

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-066901

(43)Date of publication of application : 03.03.2000

(51)Int.Cl. G06F 9/45  
G06F 9/318  
G06F 9/34

(21)Application number : 11-207155

(71)Applicant : SUN MICROSYST INC

(22)Date of filing : 22.07.1999

(72)Inventor : COX DAVID M  
MOROSOV SERGUEI V  
SEBERGER DAVID A  
WALLACE DAVID R  
WENITSKY SERGUEI L

(30)Priority

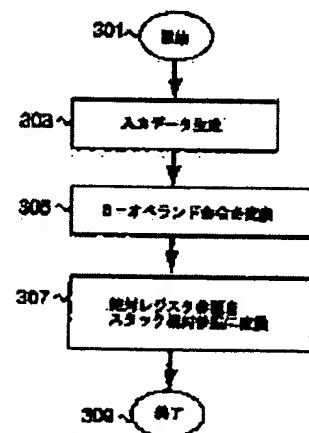
Priority number : 98 121167 Priority date : 22.07.1998 Priority country : US

#### (54) METHOD AND DEVICE FOR OPTIMIZING REGISTER IN STACK USING REGISTER ALLOCATING PART AND COMPUTER PROGRAM PRODUCT

(57)Abstract:

PROBLEM TO BE SOLVED: To optimize a calculation by using a register stack instead of a named register by converting a 3-operand instruction into an instruction less than a 3-operand.

SOLUTION: A stack register replacement process 300 is accessed in a compiler after a virtual register is allocated to a pseudo-register. An input data generation procedure 303 generates a control flow graph(CFG) of a related part of a compiled program, basic block representation related to the CFG and the intermediate representation(IR) of a compiled instruction. An instruction conversion procedure 305 converts an IR in a 3-operand format into an operand format less than a 3-operand. Next, a conversion procedure 307 converts absolute register reference into stack relative reference and converts pseudo-register address designation used in an operand of an IR instruction into register stack relative address designation.



BEST AVAILABLE COPY

## LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or

(51) IntCl. <sup>7</sup>	識別記号	F I	テーマコード(参考)
G 0 6 F 9/45		G 0 6 F 9/44	3 2 2 H
9/318		9/34	3 4 0 A
9/34	3 4 0	9/30	3 2 0 A
		9/44	3 2 2 F

審査請求 未請求 請求項の数44 O L (全 27 頁)

(21) 出願番号	特願平11-207155	(71) 出願人	595034134 サン・マイクロシステムズ・インコーポ レイテッド Sun Microsystems, I nc. アメリカ合衆国 カリフォルニア州 94303 バロ アルト サン アントニオ ロード 901
(22) 出願日	平成11年7月22日(1999.7.22)	(72) 発明者	ディビッド・エム・コックス アメリカ合衆国カリフォルニア州94550・ リバーモア・レグラスロード 573
(31) 優先権主張番号	0 9 / 1 2 1 1 6 7	(74) 代理人	100089266 弁理士 大島 陽一
(32) 優先日	平成10年7月22日(1998.7.22)		
(33) 優先権主張国	米国 (U S)		

最終頁に続く

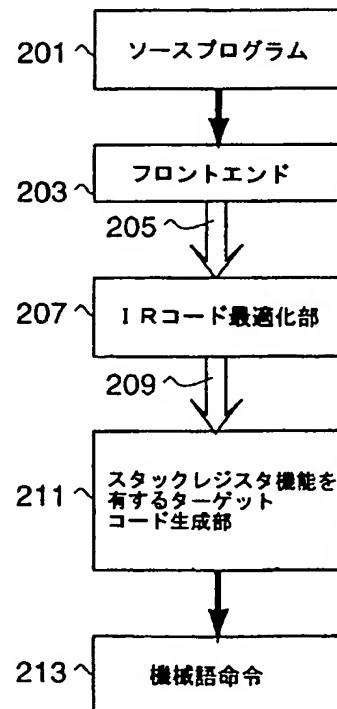
(54) 【発明の名称】 レジスタ割当て部を用いてスタック内のレジスタを最適化するための方法、装置及びコンピュータプログラムプロダクト

## (57) 【要約】

【課題】 従来のレジスタ割当て技術及び最適化技術をレジスタスタックとして構成されたレジスタに適用する方法、装置及びプログラムプロダクトを実現する。

【解決手段】 コンパイラがレジスタスタックのスタックレジスタにアクセスするための効率的なコードを生成することができる装置、方法及びコンピュータプログラムプロダクトが開示される。本発明はコンパイラの間表現内の3-オペランド命令を、1つ或いはそれ以上の3-オペランド未満の命令に変換することにより動作する。また本発明は命令のオペランドアドレス指定を、疑似名前付きレジスタへのアクセスから、レジスタスタックへのスタックオフセットを用いるスタックレジスタへのアクセスに変換する。また本発明はレジスタスタック入替えに応じて各命令においてレジスタスタック状態を判定し、それに応じてスタックレジスタへの各後続アクセスに対してスタックオフセットをマップする。

200 →



## 【特許請求の範囲】

【請求項1】 ターゲットコンピュータアーキテクチャ内に複数のスタックレジスタを備えるレジスタスタックを効率的に用いるためのコンピュータ制御による方法であって、前記方法が、

(a) 3-オペランド命令を1つ或いはそれ以上の3-オペランド未満の命令に変換する過程と、

(b) 前記1つ或いはそれ以上の3-オペランド未満の命令を解析し、1つ或いはそれ以上の疑似レジスタに対する存続期間情報を判定する過程と、

(c) 前記存続期間情報に応じて、ポップ動作を、前記1つ或いはそれ以上の3-オペランド未満の命令に挿入する過程と、

(d) オペランド参照を、前記疑似レジスタから前記レジスタスタックへのオフセットに変換し、前記複数のスタックレジスタを参照する過程とを有することを特徴とするコンピュータ制御による方法。

【請求項2】 前記ターゲットコンピュータアーキテクチャがインテル互換性浮動小数点ユニットを備えることを特徴とする請求項1に記載のコンピュータ制御による方法。

【請求項3】 前記(d)過程がさらに、

(d1) 前記疑似レジスタを前記レジスタスタックにマップする過程とをさらに有することを特徴とする請求項1に記載のコンピュータ制御による方法。

【請求項4】 前記(d1)過程がさらに、

(d1a) 第1の基本ブロックの開始時点でレジスタスタック状態を初期化する過程と、

(d1b) 前記第1の基本ブロックの1つ或いはそれ以上の命令を処理し、前記疑似レジスタに対する参照を、前記複数のスタックレジスタにアクセスするための前記レジスタスタックのオフセットに置き換える過程と、

(d1c) 必要に応じて、前記第1の基本ブロック内に1つ或いはそれ以上のレジスタスタック入替え命令を挿入する過程と、

(d1d) (d1b)過程及び(d1c)過程に応じて、前記レジスタスタック状態を更新する過程とを有することを特徴とする請求項3に記載のコンピュータ制御による方法。

【請求項5】 (d2) 前記第1の基本ブロックからの出口には存在せず、第2の基本ブロックへの入口において現存するレジスタである一組の新規の現存レジスタを判定する過程と、

(d3) 前記第2の基本ブロックへの入口では存在せず、前記第1の基本ブロックからの出口において現存するレジスタである一組の新規の無効レジスタを判定する過程と、

(d4) 前記レジスタスタックからの前記一組の新規の無効レジスタを除去し、前記レジスタスタックに前記一組の新規の現存レジスタを加えることにより、前記第1

の基本ブロックと前記第2の基本ブロックとの間のレジスタ使用を正規化する過程とをさらに有することを特徴とする請求項4に記載のコンピュータ制御による方法。

【請求項6】 (d5) 前記第1の基本ブロックと前記第2の基本ブロックとの間に正規化基本ブロックを挿入する過程と、

(d6) 前記正規化基本ブロック内に1つ或いはそれ以上の正規化命令を挿入し、前記第1の基本ブロックと前記第2の基本ブロックとの間のレジスタ使用を正規化する過程とをさらに有することを特徴とする請求項5に記載のコンピュータ制御による方法。

【請求項7】 (d5) 前記第1の基本ブロック内に1つ或いはそれ以上の正規化命令を挿入し、前記第1の基本ブロックと前記第2の基本ブロックとの間のレジスタ使用を正規化する過程とをさらに有することを特徴とする請求項5に記載のコンピュータ制御による方法。

【請求項8】 (d5) 前記第2の基本ブロック内に1つ或いはそれ以上の正規化命令を挿入し、前記第1の基本ブロックと前記第2の基本ブロックとの間のレジスタ使用を正規化する過程とをさらに有することを特徴とする請求項5に記載のコンピュータ制御による方法。

【請求項9】 複数のスタックレジスタを有するレジスタスタックを備えるターゲットコンピュータアーキテクチャに配向されるターゲットプログラムを最適化するためのコンピュータ制御による方法であって、

(a) 前記複数のスタックレジスタに1つ或いはそれ以上の疑似レジスタを割り当てる過程と、

(b) 前記割り当てられた疑似レジスタを、複数のスタックオフセットを用いて前記複数のスタックレジスタにマップするレジスタスタック状態を保守する過程であって、前記レジスタスタック状態が前記スタックレジスタの変更に応答する、該過程と、

(c) 前記割り当てられた疑似レジスタの1つを参照する命令を変換し、前記複数のスタックオフセットの1つを用いて、前記レジスタスタック状態により前記複数のスタックレジスタの1つを識別する過程とを有することを特徴とするコンピュータ制御による方法。

【請求項10】 前記ターゲットコンピュータアーキテクチャがインテル互換性浮動小数点ユニットを備えることを特徴とする請求項9に記載の方法。

【請求項11】 前記(c)過程がさらに、3-オペランド命令を1つ或いはそれ以上の3-オペランド未満の命令に変換する過程とを有することを特徴とする請求項9に記載のコンピュータ制御による方法。

【請求項12】 (d) 第1の基本ブロックからの出口において存在する前記レジスタスタックを正規化し、第2の基本ブロックへの入口において存在する前記レジスタスタックに一致させる過程とをさらに有することを特徴とする請求項9に記載のコンピュータ制御による方法。

【請求項13】 (d1) 前記第1の基本ブロックに1つ或いはそれ以上の正規化命令を加える過程をさらに有することを特徴とする請求項12に記載のコンピュータ制御による方法。

【請求項14】 (d1) 前記第2の基本ブロックに1つ或いはそれ以上の正規化命令を加える過程をさらに有することを特徴とする請求項12に記載のコンピュータ制御による方法。

【請求項15】 (d1) 前記第1の基本ブロックと前記第2の基本ブロックとの間に正規化基本ブロックを加える過程と、

(d2) 前記正規化基本ブロックに1つ或いはそれ以上の正規化命令を加える過程とをさらに有することを特徴とする請求項12に記載のコンピュータ制御による方法。

【請求項16】 複数のスタックレジスタを有するレジスタスタックを用いるターゲットコンピュータアーキテクチャに対するターゲットプログラムをコンパイルするために、中央処理装置(CPU)及び前記CPUに接続されるメモリを備える装置であって、

3-オペランド命令を1つ或いはそれ以上の3-オペランド未満の命令に変換するために構成される命令変換機構と、

前記1つ或いはそれ以上の3-オペランド未満の命令に含まれるオペランド参照を、1つ或いはそれ以上の疑似レジスタから前記レジスタスタックのオフセットに変換し、前記複数のスタックレジスタを参照するために構成される参照変換機構とを備えることを特徴とする装置。

【請求項17】 前記ターゲットコンピュータアーキテクチャがインテル互換性浮動小数点ユニットを備えることを特徴とする請求項16に記載の装置。

【請求項18】 前記参照変換機構がさらに、第1の基本ブロック内に現存する前記疑似レジスタについての情報を収集するために構成される情報収集機構と、

前記1つ或いはそれ以上の3-オペランド未満の命令を調整し、前記レジスタスタック内の現存値を保守するために構成される調整機構と、

前記1つ或いはそれ以上の3-オペランド未満の命令により参照される前記疑似レジスタを前記レジスタスタックにマップするために構成されるレジスタマップ機構とを備えることを特徴とする請求項16に記載の装置。

【請求項19】 前記レジスタマップ機構が、前記第1の基本ブロックの開始時点でレジスタスタック状態を初期化するために構成されるスタック状態初期化機構と、

前記第1の基本ブロックの前記1つ或いはそれ以上の3-オペランド未満の命令を処理し、前記疑似レジスタへの参照を、前記レジスタスタック状態を用いて前記複数のスタックレジスタにアクセスするための前記レジスタ

スタックのオフセットに置き換えるために構成される命令処理機構と、

必要に応じて前記第1の基本ブロック内に1つ或いはそれ以上のレジスタスタック入替え命令を挿入するために構成されるレジスタスタック入替え機構と、

前記命令処理機構及び前記レジスタスタック入替え機構に対応して、前記レジスタスタック状態を更新するために構成されるレジスタスタック状態メンテナンス機構とを備えることを特徴とする請求項18に記載の装置。

10 【請求項20】 前記第1の基本ブロックからの出口において存在せず、第2の基本ブロックへの入口において現存するレジスタである一組の新規の現存レジスタを判定するために構成される現存レジスタセット判定機構と、

前記第2の基本ブロックへの入口には存在せず、前記第1の基本ブロックからの出口において存在するレジスタである一組の新規の無効レジスタを判定するために構成される無効レジスタセット判定機構と、

20 前記レジスタスタックからの前記一組の新規の無効レジスタを除去し、前記レジスタスタックに前記一組の新規の現存レジスタを加えることにより、前記第1の基本ブロックと前記第2の基本ブロックとの間のレジスタ使用を正規化するために構成される正規化機構とを備えることを特徴とする請求項19に記載の装置。

【請求項21】 前記第1の基本ブロックと前記第2の基本ブロックとの間に正規化基本ブロックを挿入するために構成される基本ブロック挿入機構と、

前記正規化基本ブロック内に1つ或いはそれ以上の正規化命令を挿入し、前記第1の基本ブロックと前記第2の基本ブロックとの間のレジスタ使用を正規化するために構成される正規化命令挿入機構とをさらに備えることを特徴とする請求項20に記載の装置。

30 【請求項22】 前記第1の基本ブロック内に1つ或いはそれ以上の正規化命令を挿入し、前記第1の基本ブロックと前記第2の基本ブロックとの間のレジスタ使用を正規化するために構成される正規化命令挿入機構をさらに備えることを特徴とする請求項20に記載の装置。

【請求項23】 前記第2の基本ブロック内に1つ或いはそれ以上の正規化命令を挿入し、前記第1の基本ブロックと前記第2の基本ブロックとの間のレジスタ利用を正規化するように構成される正規化命令挿入機構をさらに備えることを特徴とする請求項20に記載の装置。

40 【請求項24】 複数のスタックレジスタを有するレジスタスタックを備えるターゲットコンピュータアーキテクチャに配向されるターゲットプログラムを最適化するために、中央処理装置(CPU)及び前記CPUに接続されるメモリを備える装置であって、前記装置が、前記複数のスタックレジスタに1つ或いはそれ以上の疑似レジスタを割り当てるために構成されるレジスタ割当て機構と、

複数のスタックオフセットを用いて前記複数のスタックレジスタに前記割り当てられた疑似レジスタをマップするレジスタスタック状態を保守するために構成されるメンテナンス機構であって、前記レジスタスタック状態が前記レジスタスタックの変更に応答する、該メンテナンス機構と、

前記割り当てられた疑似レジスタの1つを参照する命令を変換し、前記複数のスタックオフセットの1つを用いて、前記レジスタスタック状態により前記複数のスタックレジスタの1つを識別するために構成される命令参照変換機構とを備えることを特徴とする装置。

【請求項25】 前記ターゲットコンピュータアーキテクチャがインテル互換性浮動小数点ユニットを備えることを特徴とする請求項24に記載の装置。

【請求項26】 前記命令参照変換機構がさらに、3-オペランド命令を1つ或いはそれ以上の3-オペランド未満の命令に変換するために構成される命令変換機構を備えることを特徴とする請求項24に記載の装置。

【請求項27】 第1の基本ブロックからの出口において存在する前記レジスタスタックを正規化し、第2の基本ブロックへの入口において存在する前記レジスタスタックに一致させるように構成される正規化機構をさらに備えることを特徴とする請求項24に記載の装置。

【請求項28】 前記第1の基本ブロックの1つ或いはそれ以上の正規化命令を加えるために構成される正規化命令挿入機構をさらに備えることを特徴とする請求項27に記載の装置。

【請求項29】 前記第2の基本ブロックに1つ或いはそれ以上の正規化命令を加えるために構成される正規化命令挿入機構をさらに備えることを特徴とする請求項27に記載の装置。

【請求項30】 前記第1の命令ブロックと前記第2の命令ブロックとの間に正規化基本ブロックを加えるために構成される基本ブロック挿入機構と、前記正規化基本ブロックに1つ或いはそれ以上の正規化命令を加えるために構成される正規化命令挿入機構とをさらに備えることを特徴とする請求項27に記載の装置。

【請求項31】 コンピュータプログラムプロダクトであって、複数のスタックレジスタを有するレジスタスタックを用いるターゲットコンピュータアーキテクチャに対するターゲットプログラムをコンピュータがコンパイルできるようにするために与えられるコンピュータ読取り可能コードを有するコンピュータ利用可能記憶媒体を備え、前記コンピュータ読取り可能コードが、3-オペランド命令を1つ或いはそれ以上の3-オペランド未満の命令に変換するために構成される命令変換機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記1つ或いはそれ以上の3-オペランド未満の命令のオペランド参照を、1つ或いはそれ以上の疑似レジスタから前記レジスタスタックのオフセットに変換し、前記複数のスタックレジスタを参照するために構成される参照変換機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとを有することを特徴とするコンピュータプログラムプロダクト。

【請求項32】 前記参照変換機構がさらに、第1の基本ブロック内に存在する前記疑似レジスタについての情報を収集するために構成される情報収集機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、前記1つ或いはそれ以上の3-オペランド未満の命令を調整し、前記レジスタスタック内の現存値を保守するために構成される調整機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、前記1つ或いはそれ以上の3-オペランド未満の命令により参照される前記疑似レジスタを前記レジスタスタックにマップするために構成されるレジスタマップ機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとを備えることを特徴とする請求項31に記載のコンピュータプログラムプロダクト。

【請求項33】 前記レジスタマップ機構が、前記第1の基本ブロックの開始時点でレジスタスタック状態を初期化するために構成されるスタック状態初期化機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、前記第1の基本ブロック内の前記1つ或いはそれ以上の3-オペランド未満の命令を処理し、前記疑似レジスタへの参照を、前記レジスタスタック状態を用いて前記複数のスタックレジスタにアクセスするための前記レジスタスタックのオフセットに置き換えるために構成される命令処理機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

必要に応じて、前記第1の基本ブロック内に1つ或いはそれ以上のレジスタスタック入替え命令を挿入するために構成されるレジスタスタック入替え機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、前記命令処理機構及び前記レジスタスタック入替え機構に対応して、前記レジスタスタック状態を更新するために構成されるレジスタスタック状態メンテナンス機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとを有することを特徴とする請求項32に記載のコンピュータ

ログラムプロダクト。

【請求項34】 前記第1の基本ブロックからの出口において存在せず、第2の基本ブロックへの入口において現存するレジスタである一組の新規の現存レジスタを判定するために構成される現存レジスタセット判定機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、前記第2の基本ブロックへの入口には存在せず、前記第1の基本ブロックからの出口において現存するレジスタである一組の新規の無効レジスタを判定するために構成される無効レジスタセット判定機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、前記レジスタスタックからの前記一組の新規の無効レジスタを除去し、前記レジスタスタックに前記一組の新規の現存レジスタを加えることにより、前記第1の基本ブロックと前記第2の基本ブロックとの間のレジスタ使用を正規化するために構成される正規化機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとをさらに備えることを特徴とする請求項33に記載のコンピュータプログラムプロダクト。

【請求項35】 前記第1の基本ブロックと前記第2の基本ブロックとの間に正規化基本ブロックを挿入するために構成される基本ブロック挿入機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、前記正規化基本ブロック内に1つ或いはそれ以上の正規化命令を挿入し、前記第1の基本ブロックと前記第2の基本ブロックとの間のレジスタ使用を正規化するために構成される正規化命令挿入機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとを備えることを特徴とする請求項34に記載のコンピュータプログラムプロダクト。

【請求項36】 コンピュータプログラムプロダクトであって、複数のスタックレジスタを有するレジスタスタックを備えるターゲットコンピュータアーキテクチャに配向されるターゲットプログラムをコンピュータが最適化できるようにするために与えられるコンピュータ読取り可能コードを有するコンピュータ利用可能記憶媒体を備え、前記コンピュータ読取り可能コードが、前記複数のスタックレジスタに1つ或いはそれ以上の疑似レジスタを割り当てるために構成されるレジスタ割当て機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、複数のスタックオフセットを用いて前記複数のスタックレジスタに前記割り当てられた疑似レジスタをマップするレジスタスタック状態を保守するために構成されるメ

ンテナンス機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードであって、前記レジスタスタック状態が前記スタックレジスタの変更に応答する、該コンピュータ読取り可能プログラムコードと、

前記割り当てられた疑似レジスタの1つを参照する命令を変換し、前記複数のスタックオフセットの1つを用いて、前記レジスタスタック状態により前記複数のスタックレジスタの1つを識別するために構成される命令参照変換機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとを有することを特徴とするコンピュータプログラムプロダクト。

【請求項37】 前記命令参照変換機構がさらに、3-オペランド命令を1つ或いはそれ以上の3-オペランド未満の命令に変換するために構成される命令変換機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードを備えることを特徴とする請求項36に記載のコンピュータプログラムプロダクト。

【請求項38】 第1の基本ブロックからの出口において存在する前記レジスタスタックを正規化し、第2の基本ブロックへの入口において存在する前記レジスタスタックに一致させるために構成される正規化機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードをさらに備えることを特徴とする請求項36に記載のコンピュータプログラムプロダクト。

【請求項39】 前記第1の基本ブロックに1つ或いはそれ以上の正規化命令を加えるために構成される正規化命令挿入機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードをさらに備えることを特徴とする請求項38に記載のコンピュータプログラムプロダクト。

【請求項40】 前記第2の基本ブロックに1つ或いはそれ以上の正規化命令を加えるために構成される正規化命令挿入機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードをさらに備えることを特徴とする請求項38に記載のコンピュータプログラムプロダクト。

【請求項41】 前記第1の基本ブロックと前記第2の基本ブロックとの間に正規化基本ブロックを加えるために構成される基本ブロック挿入機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、前記正規化基本ブロックに1つ或いはそれ以上の正規化命令を加えるために構成される正規化命令挿入機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとをさらに備えることを特徴とする請求項38に記載のコンピュ

タプログラムプロダクト。

【請求項42】 ターゲットコンピュータアーキテクチャ内に複数のスタックレジスタを備えるレジスタスタックを効率的に使用するためのコンピュータ制御による方法であって、前記方法により、固定レジスタ名を有する命令を生成するために用いることができる技術が、前記複数のスタックレジスタを用いる命令を生成するために用いることができ、また前記方法が、

(a) プッシュ或いはポップ動作を用いずに複数の命令を生成する過程であって、前記複数の命令のオペランドが、前記ターゲットコンピュータアーキテクチャにおいて実際に実装される前記複数のスタックレジスタではなく、固定名を有する1つ或いはそれ以上の疑似レジスタである、該生成過程と、

(b) 前記複数の命令を、前記ターゲットコンピュータアーキテクチャの命令に対応するオペランドフォーマットを有する複数の新規の命令に変換する過程であって、それにより例えば、2つのソースオペランド及び1つのターゲットオペランドを有する3-オペランド命令が共有ソース及びターゲットを有する2-オペランド命令に変換される、該変換過程と、

(c) 前記複数の新規の命令を解析し、前記1つ或いはそれ以上の疑似レジスタに対する存続期間情報を判定する過程と、

(d) 前記存続期間情報を用いて、1つ或いはそれ以上のポップ動作を前記複数の新規の命令に導入し、それにより無効な値を削除する過程と、

(e) 前記1つ或いはそれ以上の疑似レジスタから前記複数のスタックレジスタへのマッピング手段を初期化する過程と、

(f) 前記複数の新規の各命令に対してコンピュータ制御によるステップを実行する過程であって、前記ステップが、

(1) 前記新規の命令内で、前記1つ或いはそれ以上の疑似レジスタを、前記マッピング手段を用いることにより1つ或いはそれ以上の前記複数のスタックレジスタに置き換える過程と、

(2) 前記マッピング手段を更新し、前記スタック効果を前記新規の命令の前記マッピングに反映させる過程とを有する、該過程とを有し、

それにより固定レジスタ名を有する命令を生成するために用いることができる前記技術を用いて、前記複数のスタックレジスタを用いる命令を生成できることを特徴とするコンピュータ制御による方法。

【請求項43】 ターゲットコンピュータアーキテクチャ内に複数のスタックレジスタを備えるレジスタスタックを効率的に使用するために、中央処理装置(CPU)及び前記CPUに接続されるメモリを備える装置であって、前記装置により、固定レジスタ名を有する命令を生成するために用いることができる技術が、前記複数の

のスタックレジスタを用いる命令を生成するために用いることができ、また前記装置が、

プッシュ或いはポップ動作を用いずに複数の命令を生成するために構成される命令生成機構であって、前記複数の命令のオペランドが、前記ターゲットコンピュータアーキテクチャ内に実際に実装される前記複数のスタックレジスタではなく、固定名を有する1つ或いはそれ以上の疑似レジスタである、該命令生成機構と、

前記複数の命令を、前記ターゲットコンピュータアーキテクチャの命令に対応するオペランドフォーマットを有する複数の新規の命令に変換するために構成される命令変換機構であって、それにより例えば2つのソースオペランドと1つターゲットオペランドを有する3-オペランド命令が共有ソース及びターゲットを有する2-オペランド命令に変換される、該命令変換機構と、前記複数の新規の命令を解析し、前記1つ或いはそれ以上の疑似レジスタに対する存続期間情報を判定するために構成される命令解析機構と、

前記存続期間情報を用いて、1つ或いはそれ以上のポップ動作を前記複数の新規の命令に導入し、それにより無効な値を削除するために構成される無効値削除機構と、前記1つ或いはそれ以上の疑似レジスタから前記複数のスタックレジスタへのマッピング手段を初期化するために構成される初期化機構と、

前記複数の新規の各命令を変換するために構成される疑似レジスタ変換機構とを備え、前記疑似レジスタ変換機構が、

前記新規の命令内において、前記1つ或いはそれ以上の疑似レジスタを前記マッピング手段を用いることにより

1つ或いはそれ以上の前記複数のスタックレジスタに置き換えるために構成されるオペランド置換え機構と、

前記マッピング手段を更新し、前記スタック効果を前記新規の命令の前記マッピングに反映させるために構成される調整機構とを備え、

それにより前記装置が、前記複数のスタックレジスタを用いる命令を生成するために、固定レジスタ名を有する命令を生成するために用いることができる前記技術を組み込むことを特徴とする装置。

【請求項44】 固定レジスタ名を有する命令を生成するために用いることができる技術が、複数のスタックレジスタを用いる命令を生成するために用いられるようにするコンピュータプログラムプロダクトであって、

コンピュータがターゲットコンピュータアーキテクチャ内に前記複数のスタックレジスタを備えるレジスタスタックを効率的に用いることができるようにするために与えられるコンピュータ読取り可能コードを有するコンピュータ利用可能記憶媒体であって、前記コンピュータ読取り可能コードが、

プッシュ或いはポップ動作を用いずに複数の命令を生成するために構成される命令生成機構を前記コンピュータ



が実現できるようにするために構成されるコンピュータ読取り可能プログラムコードであって、前記複数の命令のオペランドが、前記ターゲットコンピュータアーキテクチャ内に実際に実装される前記スタックレジスタではなく、固定名を有する1つ或いはそれ以上の疑似レジスタである、該コンピュータ読取り可能プログラムコードと、

前記複数の命令を、前記ターゲットコンピュータアーキテクチャの命令に対応するオペランドフォーマットを有する複数の新規の命令に変換するために構成される命令変換機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードであって、それにより例えば2つのソースオペランドと1つのターゲットオペランドとを有する3-オペランド命令が、共有ソース及びターゲットを有する2-オペランド命令に変換される、該コンピュータ読取り可能プログラムコードと、

前記複数の新規の命令を解析し、前記1つ或いはそれ以上の疑似レジスタに対する存続期間情報を判定するために構成される命令解析機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記存続期間情報を用いて、1つ或いはそれ以上のポップ動作を前記複数の新規の命令に導入し、それにより無効な値を削除するために構成される無効値削除機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記1つ或いはそれ以上の疑似レジスタから前記複数のスタックレジスタへのマッピング手段を初期化するために構成される初期化機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記複数の新規の各命令を変換するために構成される疑似レジスタ変換機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとを有し、前記疑似レジスタ変換機構が、前記新規の命令内において、前記1つ或いはそれ以上の疑似レジスタを前記マッピング手段を用いて1つ或いはそれ以上の前記複数のスタックレジスタに置き換えるために構成されるオペランド置換え機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードと、

前記マッピング手段を更新し、前記スタック効果を前記新規の命令の前記マッピングに反映させるために構成される調整機構を前記コンピュータが実現できるようにするために構成されるコンピュータ読取り可能プログラムコードとを備え、

それにより前記プロダクトが、前記複数のスタックレジスタを用いる命令を生成するために、固定レジスタ名を有する命令を生成するために用いることができる前記技

術を組み込むことを特徴とするコンピュータプログラムプロダクト。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明はコンピュータシステム用の最適化コンパイラの分野に関連する。詳細には、本発明は名前付レジスタの代わりに、レジスタスタックを用いて計算を最適化するための新規で、有用な最適化方法、装置及びコンピュータプログラムプロダクトに関連する。

【0002】

【従来の技術】最近の多くのコンピュータアーキテクチャは、特定レジスタを識別するために名前を用いる。最適化コンパイラは、そのような名前付レジスタの使用を最適化するための多くの技術を発展させた。いくつかのコンピュータアーキテクチャでもレジスタに対してレジスタスタックを用いる。インテル製プロセッサは、浮動小数点演算に対してレジスタスタックを用いる。このレジスタスタック構成は、Intel Architecture Software Developer's Manual: Basic Architecture (order number 243190, 1996, 1997) の第7章に記載されており、その部分を参照して本明細書の一部としている。

【0003】インテル製FPUに対する浮動小数点最適化技術は、Inter Architecture Optimization Manual (order number 242816-003, 1996, 1997) の第5章に記載されており、その部分を参照して本明細書の一部としている。

【0004】コンパイラレジスタ割当て技術は、固定名を有し、スタックレジスタを用いないレジスタを取り扱うことにより主に発展した。演算がレジスタスタック上で実行される際に、スタックレジスタ内に保持される値がレジスタスタック内を移動するため、この既存の技術はレジスタスタックを取り扱わない。

【0005】

【発明が解決しようとする課題】固定名レジスタを取り扱う従来のレジスタ割当て技術及び最適化技術を、レジスタスタックとして構成されたレジスタに適用する方法、装置及びプログラムプロダクトを実現する。

【0006】

【課題を解決するための手段】本発明は、レジスタスタックを含むターゲットコンピュータに配向されるコンピュータ命令を生成する最適化コンパイラを改善する。本発明は、コンパイラの最適化段階の多くの場合に、固定名を有する疑似レジスタを用いる方法を開示する。その後段階では、疑似レジスタを用いる命令は変更され、代わりにスタックレジスタが用いられる。スタック動作のプッシュ及びポップを実行する命令の動作により生じるレジスタスタックの入替は、疑似レジスタをスタックレジスタにマップするレジスタスタック状態により追跡される。スタックレジスタは、レジスタスタックへのオ



フセットにより参照される。

【0007】本発明の一態様は、ターゲットコンピュータアーキテクチャ内に複数のスタックレジスタを備えるレジスタスタックを効率的に用いるためのコンピュータ制御による方法である。その方法はプッシュ或いはポップ動作を用いない複数の命令を生成し、そのオペランドは、ターゲットコンピュータアーキテクチャ内に実際に実装される複数のスタックレジスタではなく、固定名を有する1つ或いはそれ以上の疑似レジスタである。またその方法は複数の命令を複数の新しい命令に変換する。複数の新しい命令は、ターゲットコンピュータアーキテクチャの命令のフォーマットに対応するオペランドフォーマットを有する。例えば、2つのソースオペランド及び1つのターゲットオペランドを有する3-オペランド命令は、共有のソース及びターゲットを有する2-オペランド命令に変換される。またその方法は複数の新しい命令を解析し、1つ或いはそれ以上の疑似レジスタに対する存続期間情報を確定する過程を含む。その方法は存続期間情報を用いて、1つ或いはそれ以上のポップ動作を複数の新しい命令に導入することにより無効な値を削除する。さらにその方法は1つ或いはそれ以上の疑似レジスタから複数のスタックレジスタへのマッピング手段を初期化する。またその方法は、複数の新しい命令のそれぞれに対して、以下のコンピュータ制御による過程を実行する。その過程は、新しい命令内でマッピング手段を用いることにより、1つ或いはそれ以上の疑似レジスタを1つ或いはそれ以上の複数のスタックレジスタに置き換える過程と、マッピング手段を更新してスタック効果を新しい命令のマッピングに反映させる過程とを含む。このようにその方法は、固定レジスタ名を有する命令を生成するための既存技術を用いて、複数のスタックレジスタを用いる命令を生成する。

【0008】本発明の別の態様は、ターゲットコンピュータアーキテクチャ内に複数のスタックレジスタを備えるレジスタスタックを効率的に用いるために、中央処理装置(CPU)及びそのCPUに接続されるメモリを備える装置である。その装置は、プッシュ或いはポップ動作を用いない複数の命令を生成するために構成された命令生成機構を備え、そのオペランドは、ターゲットコンピュータアーキテクチャ内に実際に実装される複数のスタックレジスタの代わりに、固定名を有する1つ或いはそれ以上の疑似レジスタである。またその装置は、複数の命令を複数の新しい命令に変換するために構成された命令変換機構を備える。複数の新しい命令は、ターゲットコンピュータアーキテクチャの命令フォーマットに対応するオペランドフォーマットを有する。例えば2つのソースオペランド及び1つのターゲットオペランドを有する3-オペランド命令は、共有ソース及びターゲットを有する2-オペランド命令に変換される。さらにその装置は、複数の新しい命令を解析し、1つ或いはそれ以

上の疑似レジスタに対する存続期間情報を確定するために構成された命令解析機構を備える。存続期間情報を用いて1つ或いはそれ以上のポップ動作を複数の新しい命令に導入するために構成された無効値削除機構により、無効な値は削除される。さらにその装置は、1つ或いはそれ以上の疑似レジスタから複数のスタックレジスタへのマッピング手段を初期化するために構成された初期化機構を備える。またその装置は、複数の新しい命令をそれぞれ変換するために構成された疑似レジスタ変換機構を備える。疑似レジスタ変換機構はさらに、新しい命令の中でマッピング手段を用いて、1つ或いはそれ以上の疑似レジスタを1つ或いはそれ以上の複数のスタックレジスタに置き換えるために構成されたオペランド置換え機構と、マッピング手段を更新してスタック効果を新しい命令のマッピングに反映させるために構成された調整機構とを備える。このようにしてその装置は、固定レジスタ名を有する命令を生成するために用いられる既存技術を組み込み、複数のスタックレジスタを用いる命令を生成する。

【0009】本発明の別の態様はコンピュータ利用可能記憶媒体を備えるコンピュータプログラムプロダクトであり、コンピュータ利用可能記憶媒体はその中に組み込まれるコンピュータ読取り可能コードを備える。コンピュータ上で実行される際に、コンピュータ読取り可能コードにより、そのコンピュータはターゲットコンピュータアーキテクチャ内に複数のスタックレジスタを備えるレジスタスタックを効率的に使用できるようになる。コンピュータ上で実行される際に、そのコンピュータ読取り可能コードにより、コンピュータは命令変換機構、命令解析機構、無効値削除機構、初期化機構、疑似レジスタ変換機構、オペランド置換え機構及び調整機構を実現できるようになる。これらの機構はそれぞれ、上記装置に対する対応する機構と同じ機能を有する。

【0010】本発明の別の態様は、ターゲットコンピュータアーキテクチャに配向されるターゲットプログラムを最適化するためのコンピュータ制御による方法である。ターゲットコンピュータアーキテクチャは複数のスタックレジスタを備えるレジスタスタックを含む。その方法は、1つ或いはそれ以上の疑似レジスタを複数のスタックレジスタに割り当てる過程を有する。別の過程は、複数のスタックオフセットを用いて、その割り当てられた疑似レジスタを複数のスタックレジスタにマップするレジスタスタック状態を保守する過程である。レジスタスタック状態はレジスタスタックの変化に応答する。またその方法は、1つの割り当てられた疑似レジスタを参照する命令を、複数のスタックオフセットの1つを用いる命令に変換する。複数のスタックオフセットの1つを用いるために命令を変換することにより、その命令は、レジスタスタック状態により複数のスタックレジスタの1つを識別できるようになる。

【0011】本発明のさらに別の態様は、ターゲットコンピュータアーキテクチャに配向されるターゲットプログラムを最適化するために、中央処理装置（CPU）及びCPUに接続されるメモリを備える装置である。ターゲットコンピュータアーキテクチャは、複数のスタックレジスタを備えるレジスタスタックを含む。その装置は、1つ或いはそれ以上の疑似レジスタを複数のスタックレジスタに割り当てるために構成されるレジスタ割当て機構を備える。さらにその装置は、レジスタスタック状態を保守するために構成されるメンテナンス機構を備える。レジスタスタック状態は複数のスタックオフセットを用いて、その割り当てられた疑似レジスタを、複数のスタックレジスタにマップする。レジスタスタック状態は、レジスタスタックの変化にตอบสนองする。またその装置は、割り当てられた疑似レジスタの1つを参照する命令を、複数のスタックオフセットの1つを用いる命令に変換するために構成される命令参照変換機構を備える。変換された命令オペランドは、レジスタスタック状態により複数のスタックレジスタの1つを識別する。

【0012】本発明の別の態様は、コンピュータ利用可能記憶媒体を備えるコンピュータプログラムプロダクトであり、コンピュータ利用可能記憶媒体はその中に組み込まれるコンピュータ読取り可能コードを有する。コンピュータ上で実行される際に、コンピュータ読取り可能コードにより、コンピュータは、複数のスタックレジスタを備えるレジスタスタックを含むターゲットコンピュータアーキテクチャに配向されるターゲットプログラムを最適化できるようになる。コンピュータ上で実行される際に、コンピュータ読取り可能コードにより、コンピュータはレジスタ割当て機構、メンテナンス機構及び命令参照変換機構を実現することができる。これらの機構はそれぞれ、上記装置に対する対応する機構と同じ機能を有する。

【0013】本発明の別の態様は、ターゲットコンピュータアーキテクチャ内に複数のスタックレジスタを備えるレジスタスタックを効率的に用いるためのコンピュータ制御による方法である。その方法は、3-オペランド命令を、1つ或いはそれ以上の3-オペランド命令未満の命令に変換する過程を有する。またその方法は1つ或いはそれ以上の3-オペランド命令未満の命令を解析し、1つ或いはそれ以上の疑似レジスタに対する存続期間情報を判定する。さらにその方法は、存続期間情報に応じて、ポップ動作を1つ或いはそれ以上の3-オペランド命令未満の命令に挿入する過程を有する。オペランドアドレスは、割り当てられた疑似レジスタから、レジスタスタックのオフセットに変換され、複数のスタックレジスタをアドレス指定する。

【0014】本発明の別の態様は、ターゲットコンピュータアーキテクチャに対するターゲットプログラムをコンパイルするために、中央処理装置（CPU）及びCP

Uに接続されるメモリを備える装置である。ターゲットコンピュータアーキテクチャは、複数のスタックレジスタを備えるレジスタスタックを用いる。その装置は、3-オペランド命令を1つ或いはそれ以上の3-オペランド命令未満の命令に変換するために構成される命令変換機構を備える。またその装置は、1つ或いはそれ以上の疑似レジスタからの1つ或いはそれ以上の3-オペランド命令未満の命令に含まれるオペランドアドレスをレジスタスタックのオフセットに変換し、複数のスタックレジスタをアドレス指定するために構成されるアドレス変換機構を備える。

【0015】本発明のさらに別の態様は、コンピュータ利用可能記憶媒体を備えるコンピュータプログラムプロダクトであり、コンピュータ利用可能記憶媒体がその内部にコンピュータ読取り可能コードを有する。コンピュータ上で実行される際に、コンピュータ読取り可能コードにより、コンピュータはターゲットコンピュータアーキテクチャに対するターゲットプログラムをコンパイルすることができる。ターゲットコンピュータアーキテクチャは複数のスタックレジスタを備えるレジスタスタックを用いる。コンピュータ上で実行される際に、コンピュータ読取り可能コードによりコンピュータは、命令変換機構及びアドレス変換機構を実現することができる。これらの機構はそれぞれ、上記装置に対する対応する機構と同じ機能を有する。

【0016】生変数解析（live-variable analysis）の議論は、「Compilers: Principles, Techniques and Tools by Alfred VJ」（Aho, Ravi Sethi and Jeffrey D. Ullman, Addison-Wesley Publishing Co. 1988, ISBN 0-201-10088-6）の608～633ページに見出すことができ、ここで参照して本明細書の一部としている。

【0017】本発明の上記態様及び多くの他の態様は、種々の図面において例示される以下の好適の実施例の詳細な説明から当業者には明らかになるであろう。

【0018】

【発明の実施の形態】本発明はコンピュータを用いる。概ね参照番号100を付され、本発明を支援するように構成されるコンピュータのいくつかの構成要素が図1に示されており、プロセッサ101は中央処理装置（CPU）103、メモリセクション105及び入力/出力（I/O）セクション107を備えている。I/Oセクション107はキーボード109、ディスプレイユニット111、ディスク記憶ユニット113及びCD-ROMドライブユニット115に接続される。CD-ROMドライブユニット115は、CD-ROM媒体117を読み出すことができ、CD-ROM媒体117は典型的にはプログラム及びデータ119を含む。メモリセクション105は、コンピュータ100が本発明の過程を実行できるようにする実行プログラム121を含む。またそのコンピュータは、スタックオフセットを介してアド

レス指定されるいくつかのスタックレジスタを含むレジスタスタック123を備える場合もある。好適な実施例では、レジスタスタック123はインテル製浮動小数点ユニットアーキテクチャにより動作する。ディスク記憶ユニット113及びCD-ROM媒体117を備えるCD-ROMドライブユニット115はファイル記憶機構を備える。そのようなコンピュータシステムは、本発明を実現する応用例を実行することができる。図1に示されるコンピュータシステムが、本発明の各実施例に対して必要とされないこともある装置を含むことは当業者には理解されよう。詳細にはレジスタスタック123は、実行プログラム121を含む同じコンピュータ内に備えられる必要はない。

【0019】図2は概ね参照番号200により示され、本発明を用いるコンパイラアーキテクチャを示す。コンパイラアーキテクチャ200により実装されるコンパイラは、コンパイラフロントエンドセグメント203によりターゲットプログラムのソース情報201を使用する。コンパイラフロントエンドセグメント203は、ターゲットプログラムのソース情報201に適用可能なプログラミング言語の規則によりターゲットプログラムのソース情報201の構文及び意味を処理する。コンパイラフロントエンドセグメント203は、ターゲットプログラムのソース情報201の「中間」コード表現205の少なくとも1つのバージョンを生成する。中間表現(IR)は概ね、制御フローグラフ(CFG)を表すか、或いは制御フローグラフを作成するために用いることができるデータ構造を備える。CFGを用いて、プログラムの基本ブロック間の制御の流れを表す。基本ブロックは、制御フロー構造、関数、手順或いは基本ブロック内の実行の流れを変更する他の構成体を全く含まない。基本ブロックは、1つの入口点及び1つの出口点のみを備える。

【0020】その後、「中間」コード表現205は、中間表現最適化部セグメント207により最適化される。中間表現最適化部セグメント207は、ターゲットプログラムのソース情報201の「中間」コード表現205上で動作及び調整し、当業者には周知の種々の方法によりプログラムの実行を最適化する。中間表現最適化部セグメント207は最適化済み中間表現209を生成する。コード生成部セグメント211(本発明の態様を含む)は、その最適化済み中間表現209を用いて、低レベルの最適化を実行し、物理的なレジスタを割り当て、さらにその最適化済み中間表現209からアセンブラソースコード並びにまたオブジェクトコードモジュール213を生成する。オブジェクトコードは、オブジェクトモジュールにおける2値コンピュータ命令(オペコード)を備える。アセンブラソースコードは、アセンブラソース言語における一連の記号ステートメントである。アセンブラソースコード及びオブジェクトコードのい

れも、特定のコンピュータアーキテクチャ(例えばインテル製Pentiumアーキテクチャ)を目的とする。

【0021】上記のように、コード生成部セグメント211は低レベルの最適化を実行し、オブジェクトコード(オブジェクトモジュールの形式をなす場合が多い)或いはアセンブラソースコードのいずれか(或いは両方)を生成する。プログラムの最適化済み中間表現209は概ね、固定名を有する仮想レジスタを参照する。すなわち中間表現最適化部セグメント207はターゲットコンピュータが無限の名前付き仮想レジスタを含むことを仮定する。コード生成部セグメント211の動作中に、これらの仮想レジスタはターゲットコンピュータの物理的レジスタに割り当てられる。このリソース管理は、レジスタ割当て段階或いはプロセスによりコード生成部セグメント211において実行される。レジスタ割当てプロセスの一態様は、物理的レジスタの内容がプログラム実行中の種々の時点でメモリに「流出」されることが多いが、限られた数の物理的レジスタを用いて、その種々の時点でのプログラムにより適した値を保持できるようにするというものである。そのメモリに流出した値は、プログラムが種々の実行段階まで進む際に、物理的レジスタに復元される場合が多い。レジスタ割当てプロセスは、いずれの物理的レジスタがその基本ブロックに用いられるかを判定する。

【0022】最適化コンパイラ及び関連する技術の一般的な議論は、「Compilers: Principles, Techniques and Tools by Alfred V. Aho」(Ravi Sethi and Jeffrey D. Ullman, Addison-Wesley Publishing Co. 1988, ISBN 0-201-10088-6)、詳細には463-723ページの第8、9及び10章に見い出すことができ、ここで参照して本明細書の一部としている(これ以降Ahoとよぶ)。レジスタ割当て及びメモリへの値の流出に関連するさらなる情報は、「Register Allocation & Spilling via Graph Coloring」(G.J. Chaitin, 1982, Proceedings of the SIGPLAN '82 Symp. On Compiler Construction, June 1982, pp. 98-105)により提供され、ここで参照して本明細書の一部としている。

【0023】ここではいくつかのマシンがスタックとして構成される物理的レジスタを備え、これらのレジスタは固定名を用いて参照されることができない。

【0024】本発明の一態様は、固定名を有する一組の疑似レジスタが、レジスタ割当て段階まで、さらにレジスタ割当て段階を通して用いられ、これらの疑似レジスタが、実在の物理的レジスタであるかのように取り扱われるということである。このようにしてコンパイラの中間表現最適化段階は現存する機構を用いて、その疑似レジスタが後にスタックレジスタに置き換えられるかには関係なく疑似レジスタを最適化する。このようにして従来のレジスタ割当て及び流出動作(及び多くの他のコード生成及び最適化動作)が疑似レジスタに適用される。

疑似レジスタの数はスタックレジスタの数と同じである。スタックレジスタによる疑似レジスタの実際の置換えは、コンパイラがレジスタ割当てを実行した後に呼び出される命令の状況により達成される。

【0025】好適な実施例は、インテル製FPUアーキテクチャ（或いはその相当品）を用いるターゲットコンピュータに向けられる。インテル製FPUアーキテクチャは、2-オペランドまでのオペランド及び浮動小数点レジスタスタックを用いるコンピュータ浮動小数点構成を用いる。本発明を用いて、任意のレジスタスタック内のレジスタへのアクセスを最適化することができるが、それは浮動小数点アーキテクチャに制限されないことは、添付の図面及びその説明から当業者には理解されよう。

【0026】以下の説明は本発明の動作の概要を示す。後続する説明はその動作の細部に及ぶ。これらの説明はインテル互換性FPUアーキテクチャに向けられる。ここで開示される発明は他のレジスタスタック用アーキテクチャにも適用できることは当業者には理解されよう。

【0027】図3は概ね参照番号300により示され、疑似レジスタをレジスタスタック内のスタックレジスタに割り当てるためのスタックレジスタ置換えプロセスの概要を与える。スタックレジスタ置換えプロセス300は、仮想レジスタが疑似レジスタに割り当てられた後にコンパイラにおいて呼び出される。

【0028】スタックレジスタ置換えプロセス300は、「開始」位置301で開始し、「入力データ生成」手順303に続く。「入力データ生成」手順303は、コンパイルされているプログラムの関連部分の制御フローグラフ（CFG）、CFGに関連する基本ブロック表現及びコンパイル済み命令の中間表現を生成する。入力データのいくつか或いは全部が、コンパイラの上記部分により生成されることができる。次にIRは、3-オペランド形式（概ね2ソースオペランド及び1ターゲットオペランドを有する）からのIRを、インテル製アーキテクチャの命令フォーマットに対応する3-オペランド未満のオペランド形式に変換する「命令変換」手順305により変更される。「命令変換」手順305は後に図4に関連して記載される。次に「絶対レジスタ参照をスタック相対参照に変換」手順307は、IR命令のオペランドにおいて用いられる疑似レジスタアドレス指定を、レジスタスタック相対アドレス指定に変換する。

「絶対レジスタ参照をスタック相対参照に変換」手順307は後に図6に関連して記載される。最終的にスタックレジスタ置換えプロセス300は「終了」位置309を通り終了する。

【0029】図4は概ね参照番号400により示され、IRの3-オペランド形式をより少ないオペランドを有するインテルフォーマット命令に変換するためのIR命令変換プロセスを示す。プロセス400は、図3の「命

令変換」手順305により呼び出される。プロセス400は「開始」位置401で開始し、「各基本ブロックを巡回」手順403に続く。「各基本ブロックを巡回」手順403はCFGの各基本ブロックを巡回する（繰り返す）。繰返し済み基本ブロックはそれぞれ、図5に関して後に記載される「3-オペランドを変換」手順405により処理される。各基本ブロックを巡回後、プロセス400は「終了」位置407を通り終了する。

【0030】IR命令は概ね、それぞれ異なることができる2つのソースオペランド及び1つのターゲットオペランドを有する3-オペランドフォーマットにおいて存在する。従ってインテル互換性浮動小数点アーキテクチャの場合、図5に関連して記載されるように各基本ブロックのIRは変更され、任意の3-オペランド浮動小数点IR命令は、3-オペランド未満のオペランドを備えるインテル命令フォーマットを有するIR命令に、或いは3-オペランド未満のオペランドを有する一連のIR命令に変換する。3-オペランド命令は、行先（デスティネーション）とは無関係であることができる2つのソースを識別する命令である（例えば、op R1, R2 → R3、ここでopが減算である場合には、R2はR1から引かれ、その結果がR3に格納される）。インテルフォーマット2-オペランド命令は2つのソースを用いており、ソースレジスタの1つに結果を残し、その元の値を消失させる（例えば、op R1, R2、ここでopが減算である場合には、R2はR1から引かれ、その結果がR1に残る）。いくつかの命令（例えばインテル浮動小数点平方根命令）は1つのソース上で動作し、その結果（例えばop R1）を有するソースを上書きする。IR 3-オペランド命令は、移動命令及び2-オペランド命令（例えばop R1, R2 → R3は、move R3, R1; op R3, R2に変換する）を用いることによりインテルフォーマットに変換されることができる。図5はこれらの変換がいかにして実現されるかを示す。

【0031】図5は概ね参照番号420により示され、図4の「3-オペランドを変換」手順405により呼び出されるIR命令変換プロセスを示す。プロセス420は「開始」位置421で開始し、「IR命令繰返し」手順423に続く。「IR命令繰返し」手順423は、基本ブロックの各命令を繰り返す。基本ブロックの全ての命令が処理された場合、プロセス420は「終了」位置425を通り終了する。

【0032】その基本ブロック（基本ブロックは「各基本ブロック巡回」手順403により繰り返された）の各IR命令は、IR命令の指数部を取得する「命令指数部確定」手順427により検査される。これらの指数部は命令のタイプ、IR命令により用いられるオペランド数及びソース並びにデスティネーションオペランドの位置を含む。次に「関連命令」決定手順428により「命令指数部確定」手順427により確定される指数部が検査

され、命令タイプが、IR命令が疑似レジスタの1つを参照しない（すなわちIR命令が関連しない）ということを示す場合には、プロセス420は「IR命令繰返し」手順423に戻り、次のIR命令を処理する。しかしながらIR命令が関連する（すなわち疑似レジスタの1つを参照する）場合には、プロセス420は「指数部選択」手順429に続く。

【0033】「指数部選択」手順429はIR命令オペコード及びそのオペランドを検査し、可能なIR命令変換を確定する。プログラム順（in-order）選択手順431は、IR命令が命令の第1のオペランドにその結果を格納するか否かを判定する。この条件が成り立つ場合には、プロセス420は、命令の3-オペランドを3-オペランド未満の適当なインテルフォーマットに変換する「基本変換」手順432に続く。その後プロセス420は、「IR命令繰返し」手順423の繰返しを継続する。

【0034】「実行順序変換（out-of-order）、交換可能」選択手順433は、IR命令が命令の第2のオペランドにその結果を格納するか否か、並びにその演算が交換可能であるか否かを判定する。そうである場合には、プロセス420は、命令のオペランドの順序を交換すると共に、命令の3-オペランド形式を、3-オペランド形式未満の適当なインテルフォーマットに変換する「オペランド順序交換」手順435に続く。その後プロセス420は、「IR命令繰返し」手順423の繰返しを継続する。

【0035】「実行順序変換、交換不可、両方向op」選択手順437は、IR命令が命令の第2のオペランドにその結果を格納するか否か、並びにその演算が交換不可であるか否かを判定する。そうである場合には、プロセス420は、「オペコードを逆向きオペコードに置換え」手順439に続き、その手順439が命令の3-オペランド形式を、適当な逆向き2-オペランド形式に変換する（例えば、「subtract R1, R2→R2」（R1からR2を引いて、その結果をR2に残す）は、「rs subtract R2, R1」に変換される）。その後プロセス420は「IR命令繰返し」手順423の繰返しを継続する。

【0036】「デフォルト」手順441は、上記ケースが適用されない場合に用いられる。この状況では、プロセス420は、適当なソースを行先に複製し、その後複製値にその動作を適用する「命令拡張」手順443に続く。その後プロセス420は「IR命令繰返し」手順423の繰返しを継続する。

【0037】このようにしてインテルフォーマット2-オペランド命令を用いるターゲットコンピュータアーキテクチャの場合に、プロセス420は、IR3-オペランド形式からの全ての関連する命令を、3-オペランド形式未満のインテルフォーマットに変換する。IR命令

を、インテル互換性FPUと共に用いられるFSQRT命令のような1オペランド命令に変換する方法は当業者には理解されよう。

【0038】図6は概ね参照番号500により示され、疑似レジスタを参照するIR命令オペランドを、レジスタスタックへのスタックオフセットを用いてスタックレジスタを参照するIR命令オペランドに変換するための「絶対レジスタ参照をスタック相対参照に変換」プロセスを示す。プロセス500は、図3の「絶対レジスタ参照をスタック相対参照に変換」手順307により呼び出される。プロセス500は「開始」位置501で開始し、図7に関連して後に記載されるように、疑似レジスタの使用についての存続期間情報を収集する「疑似レジスタについての情報を収集」手順503に続く。存続期間情報は、レジスタのその値が無効になる時点を判定する。次にプロセス500は関連するIR命令（すなわち疑似レジスタにアクセスする命令）を検査し、実行時に、レジスタスタックから無効レジスタを除去する命令を挿入或いは変更する。「スタックから無効レジスタ除去」手順505は図8に関連して後に記載される。次にプロセス500は、疑似レジスタを参照するオペランドを、レジスタスタック内のスタックレジスタを参照するオペランドに置き換える「疑似レジスタをスタックレジスタ上にマップ」手順507に続く。「疑似レジスタをスタックレジスタ上にマップ」手順507は図7に関連して後に記載される。プロセス500は「終了」位置509を通り終了する。

【0039】図7は概ね参照番号510により示され、図6の「疑似レジスタについての情報を収集」手順503により呼び出される、疑似レジスタ存続期間伝搬プロセスを示す。プロセス510は、CFG及び関連する基本ブロックにより表されるプログラムの疑似レジスタ存続期間を伝搬する。またプロセス510は、疑似レジスタが各基本ブロックへの入口及び基本ブロックからの出口上に存在する情報を判定及び退避する。この情報を用いて、レジスタの値が存在する領域を判定する。レジスタが存在しない場合には、「無効（dead）」と呼ばれる。存続期間解析は当分野において周知であり、「Aho」に記載される。

【0040】「CFGの疑似レジスタ最適化」手順513は、いずれの疑似レジスタが各基本ブロックの入口に存在するかを判定し、input(block\_id)としてこの情報を退避する。またこの手順513は、いずれの疑似レジスタが各基本ブロックの出口に存在するかを判定し、この情報をoutput(block\_id)として退避する。このプロセス510は「終了」位置521を通り終了する。

【0041】図8は概ね参照番号530により示され、無効疑似レジスタに関連するスタックレジスタが、無効である場合にレジスタスタックからポップされるように

関連するIR命令を変更するための「スタックから無効レジスタを除去」プロセスを示す。プロセス530は、図6の「スタックから無効レジスタを除去」手順505から呼び出され、「開始」位置531で開始する。プロセス530は、CFGの各基本ブロックを巡回する「各基本ブロック巡回」手順533に続く。プロセス530は、各基本ブロックを、疑似レジスタの1つにアクセスする基本ブロックの各命令を繰り返す「基本ブロックの各関連命令繰返し」手順535に渡す。一度基本ブロックの全ての命令が繰返された場合には、プロセス530は「各基本ブロック巡回」手順533に戻り、次の基本ブロックを処理する。「基本ブロックの各関連命令繰返し」手順535により繰返される命令は、IR命令を変更し、スタックから無効スタックレジスタを除去する「スタック安定化」手順536に渡される。これはオペレーションコードを、スタックポップを実行し、スタックから無効な値を除去するバージョンに置き換えることにより行われる。また、この時点でIR命令はレジスタスタックに直接アクセスしないということは当業者には理解されよう。代わりにIR命令は、レジスタスタック内のスタックレジスタにマップされることになる疑似レジスタにアクセスする。「スタック安定化」手順536はさらに図10に関連して記載される。全ての基本ブロックが処理された後、プロセス530は「終了」位置537を通り終了する。命令のオペランドが疑似レジスタを表し、スタックレジスタを表さない場合でも、レジスタスタックに作用するためのIR命令の変更を、この時点で行なうことができることは当業者には理解されよう。

【0042】図9は概ね参照番号550により示され、関連命令のオペランドを、疑似レジスタ参照を用いることからレジスタスタックオフセット参照を用いることに交換する「疑似レジスタアドレス指定をレジスタスタックアドレス指定に変換」プロセスを示す。またプロセス550は、CFGのエッジにより接続される基本ブロック間のレジスタスタックを正規化する。プロセス550は図6の「疑似レジスタをスタックレジスタ上にマップ」手順507により呼び出され、「開始」位置551で開始する。プロセス550は、CFGの先頭から「スタックレジスタをマップ」手順555に各基本ブロックを与えるCFGの底までCFGを通過する「先頭から底までCFGを通過」手順553に続く。「スタックレジスタをマップ」手順555は、各関連する命令のオペランドを、疑似レジスタの代わりに、レジスタスタックオフセットに変換する。「スタックレジスタをマップ」手順555はさらに図11に関連して記載される。

【0043】一度関連する命令が各基本ブロックにおいて変換されたなら、プロセス550は、CFGをその底からその先頭まで通過し、CFGの各エッジ及びそのエッジに関連する基本ブロックにおいて「基本ブロック正

規化」手順559を呼び出す「底から先頭までCFGを通過」手順557に続く。「基本ブロック正規化」手順559は、レジスタスタックを入れ替える命令を、適当な基本ブロック（或いは新規に作成された基本ブロック）に挿入することにより達成される。「基本ブロック正規化」手順559は後に図12及び図13に関連して記載される。「底から先頭までCFGを通過」手順557が終了した後、プロセス550は「終了」位置561を通り終了する。

10 【0044】上記のように、インテル互換性浮動小数点ユニットアーキテクチャは、スタックレジスタにアクセスするために1-オペランド命令或いは2-オペランド命令を用いる。これは、命令ソースの1つが命令ターゲットと同じであることを意味する。従ってソース値の1つが命令の実行により消失する。両方のソース値が命令の実行中に保存される必要がある場合には、その命令の実行により消失するようになる1つの値が一時的レジスタに複製されなければならない。他のコンピュータアーキテクチャは同様の特性を有し、同様の方法において取り扱われる。

20 【0045】図10は概ね参照番号620により示され、IR命令を変更し、レジスタスタックから無効疑似レジスタを除去するために用いられるスタック安定化プロセスを示す。IRはなおも疑似レジスタを参照するが、IR命令は、命令実行後に疑似レジスタが無効である場合には、命令のスタックポップ別形態を用いるために変更されることができる。詳細には疑似レジスタの値は最後の使用後に無効になる。

30 【0046】プロセス620は「開始」位置621で開始し、「無効値」決定手順623に続く。「無効値」決定手順623は、命令により参照される疑似レジスタが命令後に無効であるか否かを判定する。そうである場合には、プロセス620は、その命令が無効値をポップする別形態或いはシーケンスを有するか否かを判定する「命令がポップ別形態所有」決定手順626に続く。命令により参照される疑似レジスタが命令後に無効でない場合には、プロセス620は「終了」位置627を通り終了する。

40 【0047】「命令がポップ別形態所有」決定手順626が、命令がポップ別形態を有するものと判定する場合には、プロセス620は非ポップ命令別形態をポップ命令別形態に置き換える「命令別形態変更」手順629に続く。ポップ命令別形態が存在しない場合には、プロセス620はコードシーケンスを挿入し、無効疑似レジスタを明示的にポップする「ポップ命令シーケンス挿入」手順628に続く。プロセス620は「終了」位置627を通り終了する。

50 【0048】ここでレジスタスタック状態の使用について議論することは有用である。レジスタスタック状態は、疑似レジスタとスタックレジスタとの間のマッピング



グを実現する。スタックレジスタがレジスタスタックとして構成されるので、他の値がレジスタスタックへプッシュ及びレジスタスタックからポップされるのに応じて、特定の値を含むスタックレジスタへのスタックオフセットが変化する。レジスタスタック状態は疑似レジスタをスタックレジスタにマップする。本発明の一態様は、レジスタスタックが変化するのに応じて、レジスタ\*

\*スタック内の値の位置を追跡することによるレジスタスタック状態のメンテナンスである。基本ブロックの一部を構成する $x = (a+b) * (a+d)$ を考慮する。レジスタ割当て部からのIRコードは、表1と同様の内容を探索するであろう。

【0049】

【表1】

表1

f l d	a → p r 1
f s t o	p r 1 → p r 2
f a d d	p r 1, b → p r 1
f a d d	p r 2, d → p r 2
f m u l	p r 2, p r 1 → p r 2
f s t o	p r 2 → x

【0050】ここで「p r n」は疑似レジスタ「n」を示す。 ※かを示す。

20 【0052】

【0051】表2は、疑似レジスタがレジスタスタック状態を用いてスタックレジスタに如何に割り当てられる※

【表2】

表2

I n s t	O p e r a n d s	F 0	F 1	F 2	F...
f l d	a → F 0	p r 1	p r 3		?
f s t o	F 0 → F 2	p r 1	p r 3	p r 2	?
f a d d	F 0, b → F 0	p r 1	p r 3	p r 2	?
f x c h	F 2, F 0	p r 2	p r 3	p r 1	?
f a d d	F 0, d → F 0	p r 2	p r 3	p r 1	?
f m u l	F 2, F 0 → F 0	p r 2	p r 3	p r 1	?
f s t o p	F 0 → X	p r 3	p r 1	?	?
...	...	...	...	...	...

【0053】ここで「F n」はスタックレジスタ「n」を示し、「p r n」は再び疑似レジスタ「n」を示す。表2の右4段は、いずれの疑似レジスタが各スタックレジスタに割り当てられたかを示す。「p r 3」は後続の動作に用いられる現存の疑似レジスタである。「p r 2」は、「X」への格納後に無効であると見なされ、そのためレジスタスタックからポップされるということに注目されたい。レジスタスタック状態は、レジスタスタックの先頭からスタックレジスタへのオフセットを有する各疑似レジスタ間の関係を保持する。レジスタスタック状態は全ての命令に対して保持される。レジスタスタックの先頭からのオフセットを用いるために命令を変更することにより、疑似レジスタを用いる各命令のオペランドは、疑似レジスタにマップされるスタックレジスタをアドレス指定するように変更される。レジスタスタック状態に格納された各オフセットは、レジスタスタック

がその命令により入れ替えられるのに応じて変化する。

【0054】図11は概ね参照番号700により示され、疑似レジスタをレジスタスタック内のスタックレジスタにマップするための疑似レジスタマッピングプロセスを示す。プロセス700は図4の「スタックレジスタをマップ」手順555により呼び出される。プロセス700は「開始」位置701で開始し、「初期基本ブロックマッピング状態退避」手順703に続く。「初期基本ブロックマッピング状態退避」手順703は、レジスタスタック状態を初期化する。レジスタスタック状態は、スタックオフセット値を疑似レジスタに関連付けることにより、疑似レジスタをレジスタスタック内のスタックレジスタにマップする。「初期基本ブロックマッピング状態退避」手順703は、基本ブロックへの入口におけるレジスタスタック状態の複製を退避する。初期基本ブロックレジスタスタック状態が退避された後、プロセス



700は「各命令繰返し」手順705に続く。「各命令繰返し」手順705は基本ブロック内の各命令を繰返す。基本ブロック内の全ての命令が繰返された後、プロセス700は、その基本ブロックの終了時に存在するレジスタスタック状態をfstate(block-id, output)として退避する。「最終基本ブロックマッピング状態退避」手順707に続く。その後プロセス700は「終了」位置709を通り終了する。

【0055】「初期基本ブロックマッピング状態退避」手順703及び「最終基本ブロックマッピング状態退避」手順707により退避されるレジスタスタック状態を用いて、図12に関連して後に記載されるように、基本ブロック間のレジスタスタック使用を正規化する。

【0056】「各命令繰返し」手順705により繰返される各命令は検査され、「浮動小数点レジスタ命令」決定手順711においてその命令が疑似レジスタにアクセスするか否かを判定する。繰返し済み命令が疑似レジスタにアクセスしない場合には、その命令は関連せず、プロセス700は「各命令繰返し」手順705に戻り、次の命令を繰返す。

【0057】しかしながら繰返し済み命令が疑似レジスタにアクセスする場合には、プロセス700は、繰返し済み命令が、レジスタスタック（及び関連するレジスタスタック状態）に対する入替えを必要とするか否かを判定する「スタック状態変更」決定手順713に続く。1つのそのような命令の一例は、演算される値がレジスタスタックの先頭スタックレジスタ内に格納されることが必要となるインテルFSQRT命令である。従って対象の値が任意の他のスタックレジスタ内に格納される場合には、そのレジスタスタックは、その対象の値がレジスタスタックの先頭に移動するように入れ替えられなければならない。「スタック状態変更」決定手順713が、繰返し済み命令がスタック入替えを必要とすると判定する場合には、プロセス700は「レジスタスタック入替え命令挿入」手順715に続く。「レジスタスタック入替え命令挿入」手順715は、レジスタスタックを繰返し済み命令に適した状態にするために命令を挿入する。インテル互換性FPUアーキテクチャ内部では、「レジスタスタック入替え命令挿入」手順715は、レジスタスタックを入れ替えるためにFXCH命令を挿入する。次にレジスタスタック状態は「マッピング状態更新」手順717により更新される。こうしてレジスタスタック状態は、レジスタスタックのスタックレジスタの位置の変化に応答する。従ってコンパイラは、レジスタスタックにアクセスする各命令の動作に応じてレジスタスタックの状態を保持する。

【0058】「スタック状態変更」決定手順713が、繰返し済み命令がスタック入替えを必要としないと判定する場合には、プロセス700は「マッピング状態更新」手順717に続く。「マッピング状態更新」手順7

17は挿入された命令を検査し、その命令がレジスタスタックを変更する場合には、その手順がレジスタスタック状態への対応する変更を生成する。こうしてコンパイラは、レジスタスタックにアクセスする各命令の動作に応じてレジスタスタックの状態を保持する。

【0059】一度レジスタスタック状態が更新されれば、プロセス700は、レジスタスタック状態の情報を用いて、疑似レジスタからの繰返し済み命令のオペランドを、レジスタスタックへのスタックオフセットに変換する「疑似レジスタのレジスタスタックへのマップ」手順719に続く。こうして変換された命令は、疑似レジスタの代わりにレジスタスタックのスタックレジスタを参照する。一度繰返し済み命令が変換されれば、プロセス700は、基本ブロック内の全ての命令が繰返されるまで「各命令繰返し」手順705において命令を繰返す。

【0060】プロセス700は、CFGの先頭から底への通過中に各命令ブロックに対して呼び出される。CFGの先頭から底への通過の終了時に、各命令ブロックに対する入力レジスタスタック状態及び出力レジスタスタック状態が退避されている。さらに疑似レジスタを用いる基本ブロック内の各命令は、ここでレジスタスタックへのオフセットを用いてレジスタスタックにアクセスする。

【0061】残りのステップは、親基本ブロックからの出口におけるレジスタスタックの状態を、現在の基本ブロックの開始時点におけるレジスタスタックの状態に突き合わせることである。これは、「CFGの底から先頭への通過」手順557により実行されるCFGの底から先頭への通過中にブロックを正規化することにより達成される。

【0062】図12は、概ね参照番号800により示され、親基本ブロック(block-j)からの最後のレジスタスタック使用を現在の基本ブロック(block-k)の入力レジスタスタック使用に正規化する正規化命令を挿入する基本ブロックレジスタスタック正規化プロセスを示す。正規化は、いずれのレジスタが現存する入力block-kであるが、block-jからの出口には現存しないかを判定し、かついずれのレジスタがblock-jからの出口に現存するが、block-kには用いられないかを判定するプロセスである。プロセス800は、「CFGの底から先頭への通過」手順557によりCFGが底から先頭まで通過されるのに応じて、「基本ブロック正規化」手順559により呼び出される。プロセス800は「開始」位置801で開始し、「新規の現存レジスタ判定」手順803に続く。「新規の現存レジスタ判定」手順803は、親ブロックからの出口レジスタスタック状態を、現在ブロックからの入口レジスタスタック状態と比較する。こうしてCFGのblock-jとblock-kとの間の一致を探索する

場合に、新規の現存レジスタのセットは以下のように定義される。

【0063】

【数1】  $\text{push}(\text{block-j}, \text{block-k}) = \text{input}(\text{block-k}) - \text{fstate}(\text{block-j}, \text{output})$  同様に、**「親ブロックの無効レジスタ判定」**手順805は、以下に定義されるような同じエッジにおける新規の無効レジスタのセットを判定する。

【0064】

【数2】  $\text{pop}(\text{block-j}, \text{block-k}) = \text{fstate}(\text{block-j}, \text{output}) - \text{input}(\text{block-k})$

**「正規化要求」**決定手順807は、 $\text{block-j}$  出口におけるレジスタスタック状態を $\text{block-k}$  入口におけるレジスタスタック状態に突き合わせるために必要とされるスタック入替え並びにプッシュ及びポップ値を検査する。 $\text{block-j}$  と $\text{block-k}$  との間で正規化が必要とされる場合には、プロセス800は**「基本ブロック正規化」**手順809に続く。**「基本ブロック正規化」**手順809は、レジスタスタックを操作し、 $\text{block-j}$  出口レジスタスタック状態が、図13に関連して後に記載されるように $\text{block-k}$  入力レジスタスタック状態と突き合わせられるようにする命令を挿入する。その後プロセス800は**「終了」**位置811を通り終了する。

【0065】しかしながら $\text{block-j}$  と $\text{block-k}$  との間に正規化が必要とされない場合には、プロセス800は**「終了」**位置811を通り終了する。

【0066】図13は、概ね参照番号850により示され、 $\text{block-j}$  と $\text{block-k}$  との間のレジスタスタックを正規化する基本ブロックを挿入する正規化命令挿入プロセスを示す。プロセス850は、図12の**「基本ブロック正規化」**手順809により呼び出される。プロセス850は**「開始」**位置851で開始し、新規の基本ブロックがCFGに挿入されなければならないか否かを、或いは現存する基本ブロックにおいてレジスタスタック正規化が発生することができるか否かを判定する**「新規の基本ブロック要求」**決定手順853に続く。新規の基本ブロックは、 $\text{block-k}$  が $\text{block-j}$  に対する唯一の後続ブロックである場合には必要とされない。この状況では、レジスタスタック正規化命令は $\text{block-j}$  に直接挿入されることができる。また $\text{block-j}$  が $\text{block-k}$  に対する唯一の親である場合には、正規化命令は、 $\text{block-k}$  の始めに直接挿入されることができる。この状況が当てはまらない場合には、正規化基本ブロックは、**「正規化基本ブロック挿入」**手順855においてCFGに挿入される。正規化基本ブロックは、親基本ブロックから出るレジスタスタックを正規化し（入れ替え）、子基本ブロックへの入口に

おいて予想されるレジスタスタックと一致させる命令を含む。これらの状態が図14及び図15により示される。

【0067】**「レジスタスタックプッシュ命令挿入」**手順857は、レジスタスタック上に新規の現存レジスタ（上記プッシュ定義により定義された）をプッシュする命令を挿入する。次に**「レジスタスタックポップ命令挿入」**手順859は、レジスタスタックをリオーダする命令及びレジスタスタックから無効レジスタをポップする命令を挿入する。その後**「レジスタスタック入替え命令挿入」**手順861は、 $\text{block-k}$  に対する入力レジスタスタック状態に応じてレジスタスタックを配置する命令を挿入する。次にプロセス850は**「終了」**位置863を通り終了する。

【0068】図12及び図13のプロセスはエッジにより接続される基本ブロックのレジスタスタック状態を用いて、基本ブロック間のレジスタスタックを正規化することは当業者には理解されよう。この正規化はレジスタスタックを入れ替え、レジスタスタックの現存のレジスタが、基本ブロックにおいて使用するためのレジスタスタックの適当なオフセット位置に存在できるようにする過程を含む。

【0069】図14は、概ね参照番号900により示され、本発明により動作することができる制御フローグラフを示す。制御フローグラフ900は、エッジ-j/k 905により $\text{block-k}$  903に接続される $\text{block-j}$  901を含む。 $\text{block-k}$  903はエッジ-k/m 910を用いて $\text{block-m}$  909に接続する。さらに $\text{block-l}$  907はエッジ-l/m 911により $\text{block-m}$  909に接続する。また $\text{block-k}$  903は、エッジ-k/n 912を介して $\text{block-n}$  913に接続する。また $\text{block-l}$  907はエッジ-l/n 915を用いて $\text{block-n}$  913に接続する。 $\text{block-m}$  909はエッジ-m/o 918を用いて $\text{block-o}$  917に接続する。 $\text{block-o}$  917はエッジ-o/p 920を用いて $\text{block-p}$  919に接続する。また $\text{block-n}$  913はエッジ-n/o 921を用いて $\text{block-o}$  917に接続する。また $\text{block-o}$  917はエッジ-o/q 925を用いて $\text{block-q}$  923に接続する。また $\text{block-q}$  923はエッジ-q/r 928を用いて $\text{block-r}$  927に接続する。また $\text{block-r}$  927はエッジ-r/r 929によりそれ自体に接続する。

【0070】図14を検証する場合に以下の点に注目されたい。 $\text{block-k}$  903は $\text{block-j}$  901に隣接する。 $\text{block-o}$  917は2つの親（ $\text{block-m}$  909及び $\text{block-n}$  913）及び2つの子（ $\text{block-p}$  919及び $\text{block-q}$  923）を有する。 $\text{block-m}$  909、 $\text{block-n}$  91

3及びblock-r927はそれぞれ、ブロックの上部に入る2つのエッジを有する。

【0071】図15は、概ね参照番号950により示され、図14の制御フローグラフ900上で動作する図12及び図13により示されるプロセスの動作から生じる正規化された制御フローグラフを示す。制御フローグラフ900は、親基本ブロックを出るスタックレジスタを、子基本ブロックに入るスタックレジスタに入れ替えるための正規化命令を含む基本ブロックを挿入するように変更される。図12のプロセス800を制御フローグラフ900に適用することにより、以下のことがわかる。

【0072】1) block-k903はblock-j901に隣接する。従って図13のプロセス850により挿入されるレジスタスタック正規化命令はblock-j901(block-k903に対する親ブロック)内に挿入されることができる。以下の基本ブロックの組み合わせ、block-m909及びblock-o917、block-n913及びblock-o917並びにblock-q923及びblock-r927の場合に同様の状況が生じる。

【0073】2) block-o917はblock-p919及びblock-q923の両方に対する親ブロックである。従ってレジスタスタック正規化命令(各子ブロックに適している)は、図13のプロセス850により、block-p919及びblock-q923内に挿入される。

【0074】3) 一般的な場合に、block-k903及びblock-l907(並びにblock-r927のループエッジ)の子ブロックはそれぞれ、一般に正規化命令が任意の現存の基本ブロックに挿入されることができない(基本ブロックは実行フロー分岐を持たない)ため、親基本ブロックと子基本ブロックとの間のレジスタスタック状態を正規化するために、CFGに挿入される正規化基本ブロックを有する必要がある。従ってblock-k903とblock-m909との間のレジスタスタックを正規化するための命令を含むblock-k/m951がエッジk/m910に挿入される。block-k/m951は、block-k903を出るレジスタスタック状態により識別される構成からのレジスタスタックを変更し、block-m909に対する入力レジスタスタック状態を一致させる。block-k/n953、block-l/n955及びblock-l/m957は、それぞれのエッジに対して相応する正規化を実現する。最終的にblock-r/r959がエッジr/r929に挿入され、block-r927の出口レジスタスタック状態を、block-r927の入口レジスタスタック状態に正規化する。親出口レジスタスタック状態が、子ブロックの入口レジスタスタックに完全に一致する場合には、正規化プ

ロックはCFGに挿入される必要はないということは当業者には理解されよう。

【0075】本発明は、基本ブロック間のスタック正規化命令数を最小限にし、コード拡張を低減するということは当業者には理解されよう。

【0076】本発明により、現存するレジスタ割当て及び最適化技術が、レジスタスタック内に構成されるスタックレジスタと共に用いられるようになるということは当業者には理解されよう。

10 【0077】上記内容から、本発明は(限定するわけではないが)以下の利点を有することが明らかであろう。

【0078】1) 本発明はレジスタスタックにおけるレジスタスワッピングを最小限にする。

【0079】2) 本発明は基本ブロック間でレジスタスタック使用を正規化することから生じる、可能なコード拡張を最小限にする。

20 【0080】本発明は好適な実施例に関連して記載されてきたが、本発明の範囲から逸脱することなく種々の変更例及び代替例が実現されることは当業者には理解されよう。従って本発明の範囲はここで議論された特定の本発明の実施例に制限されるわけではなく、添付の請求項及びその等価内容によってのみ画定されるべきである。

【0081】

【発明の効果】上記のように、本発明により、従来のレジスタ割当て技術及び最適化技術を、レジスタスタックとして構成されたレジスタに適用する方法、装置及びプログラムプロダクトを実現することができる。

【図面の簡単な説明】

30 【図1】好適な実施例に従って本発明を用いることができるコンピュータシステムを示す。

【図2】好適な実施例によるコンパイラを示す。

【図3】好適な実施例によるスタックレジスタ置換えプロセスの概要を示す。

【図4】好適な実施例によるIR3-オペランド命令変換プロセスの概要を示す。

【図5】好適な実施例による詳細なIR3-オペランド命令変換プロセスを示す。

40 【図6】好適な実施例により、疑似レジスタからの命令アクセスをスタックレジスタに変換するためのプロセスを示す。

【図7】好適な実施例により、CFGにより表される現存疑似レジスタを最適化し、基本ブロックへの入口及び基本ブロックからの出口において現存疑似レジスタを確定するプロセスを示す。

【図8】好適な実施例により無効スタックレジスタをレジスタスタックから除去するためのプロセスの概要を示す。

【図9】好適な実施例により疑似レジスタをスタックレジスタにマップするためのプロセスの概要を示す。

50 【図10】好適な実施例により無効疑似レジスタを基本

ブロックから除去するために用いられるスタック安定化プロセスを示す。

【図11】好適な実施例により疑似レジスタをレジスタスタックのレジスタにマップするために用いられる詳細なプロセスを示す。

【図12】好適な実施例による基本ブロックレジスタスタック正規化プロセスの概要を示す。

【図13】好適な実施例による詳細な正規化命令挿入プロセスを示す。

【図14】好適な実施例により動作する非正規化CFG 10の一部を示す。

【図15】図14のCFGに好適な実施例を適用した結果生じる正規化制御フローグラフを示す。

#### 【符号の説明】

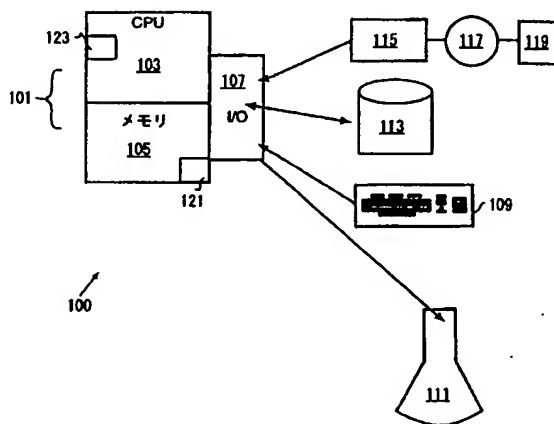
100 コンピュータ  
 101 プロセッサ  
 103 中央処理装置 (CPU)  
 105 メモリ  
 107 入力/出力 (I/O)  
 109 キーボード  
 111 ディスプレイユニット  
 113 ディスク記憶ユニット  
 115 CD-ROMドライブユニット  
 117 CD-ROM媒体  
 119 プログラム及びデータ  
 121 実行プログラム  
 123 レジスタスタック  
 200 コンパイラアーキテクチャ  
 201 ソースプログラム  
 203 フロントエンド  
 205 中間コード表現  
 207 中間表現 (IR) コード最適化部  
 209 最適化済み中間表現  
 211 コード生成部  
 213 アセンブラソース/オブジェクトコードモジュール  
 300 スタックレジスタ置換えプロセス  
 301 開始位置  
 303 「入力データ生成」手順  
 305 「3-オペランド命令変換」手順  
 307 「絶対レジスタ参照をスタック相対参照に変換」手順  
 309 終了位置  
 400 IR命令変換プロセス  
 401 開始位置  
 403 「各基本ブロック巡回」手順  
 405 「3-オペランドIR命令変換」手順  
 407 終了位置  
 420 IR命令変換プロセス  
 421 開始位置

423 「IR命令繰返し」手順  
 425 終了位置  
 427 「命令指数部確定」手順  
 428 「関連命令」決定手順  
 429 「指数部選択」手順  
 431 「プログラム順 (in-order) 選択」手順  
 432 「基本変換」手順  
 433 「実行順序変換 (out-of-order)、交換可能」  
 選択手順  
 435 「オペランド順序交換」手順  
 437 「実行順序変換、交換不可、両方向op」選択  
 手順  
 439 「オペコードを逆向きオペコードに置換え」手  
 順  
 441 「デフォルト」手順  
 443 「命令拡張」手順  
 500 「絶対レジスタ参照をスタック相対参照に変  
 換」プロセス  
 501 開始位置  
 503 「疑似レジスタについての情報収集」手順  
 505 「スタックから無効レジスタを除去」手順  
 507 「疑似レジスタをスタックレジスタ上にマッ  
 プ」手順  
 509 終了位置  
 510 疑似レジスタ存続期間伝搬プロセス  
 511 開始位置  
 513 「CFGの疑似レジスタ最適化」手順  
 521 終了位置  
 530 「スタックから無効レジスタ除去」プロセス  
 531 開始位置  
 533 「各基本ブロック巡回」手順  
 535 「基本ブロックの各関連命令繰返し」手順  
 536 「スタック安定化」手順  
 537 終了位置  
 550 「疑似レジスタアドレス指定をレジスタスタ  
 ックアドレス指定に変換」プロセス  
 551 開始位置  
 553 「先頭から底までCFGを通過」手順  
 555 「スタックレジスタをマップ」手順  
 557 「底から先頭までCFGを通過」手順  
 559 「基本ブロック正規化」手順  
 561 終了位置  
 620 スタック安定化プロセス  
 621 開始位置  
 623 「無効値」決定手順  
 626 「命令がポップ別形態所有」決定手順  
 627 終了位置  
 628 「ポップ命令シーケンス挿入」手順  
 629 「命令別形態変更」手順  
 700 疑似レジスタマッピングプロセス

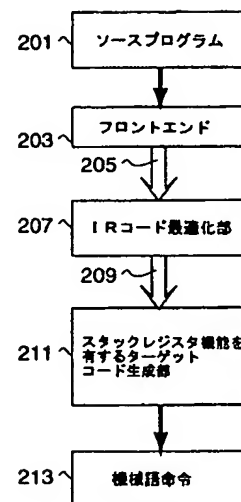
701 開始位置  
 703 「初期基本ブロックマッピング状態退避」手順  
 705 「各命令繰返し」手順  
 707 「最終基本ブロックマッピング状態退避」手順  
 709 終了位置  
 711 「浮動小数点レジスタ命令」決定手順  
 713 「スタック状態変更」決定手順  
 715 「レジスタスタック入替え命令挿入」手順  
 717 「マッピング状態更新」手順  
 719 「擬似レジスタのレジスタスタックへのマッ  
 プ」手順  
 800 基本ブロックレジスタスタック正規化プロセス  
 801 開始位置  
 803 「新規の現存レジスタ判定」手順  
 805 「親ブロックの無効レジスタ判定」手順  
 807 「正規化要求」決定手順  
 809 「基本ブロック正規化」手順  
 811 終了位置  
 850 正規化命令挿入プロセス  
 851 開始位置  
 853 「新規の基本ブロック要求」決定手順  
 855 「正規化ブロック挿入」手順  
 857 「レジスタスタックプッシュ命令挿入」手順  
 859 「レジスタスタックポップ命令挿入」手順  
 861 「レジスタスタック入替え命令挿入」手順  
 863 終了位置

900 制御フローグラフ  
 901 block-j  
 903 block-k  
 905 エッジ-j/k  
 907 block-l  
 909 block-m  
 910 エッジ-k/m  
 911 エッジ-l/m  
 912 エッジ-k/n  
 913 block-n  
 915 エッジ-l/n  
 917 block-o  
 918 エッジ-m/o  
 919 block-p  
 920 エッジ-o/p  
 921 エッジ-n/o  
 923 block-q  
 925 エッジ-o/q  
 927 block-r  
 928 エッジ-q/r  
 929 エッジ-r/r  
 951 block-k/m  
 953 block-k/n  
 955 block-l/n  
 957 block-l/m  
 959 block-r/r

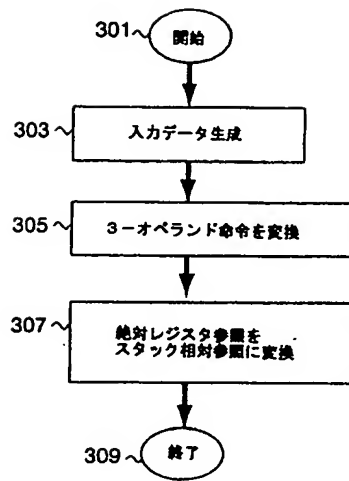
【図1】



【図2】

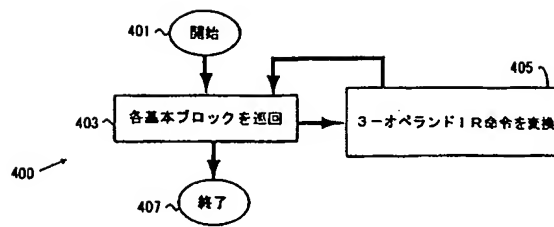


【図3】

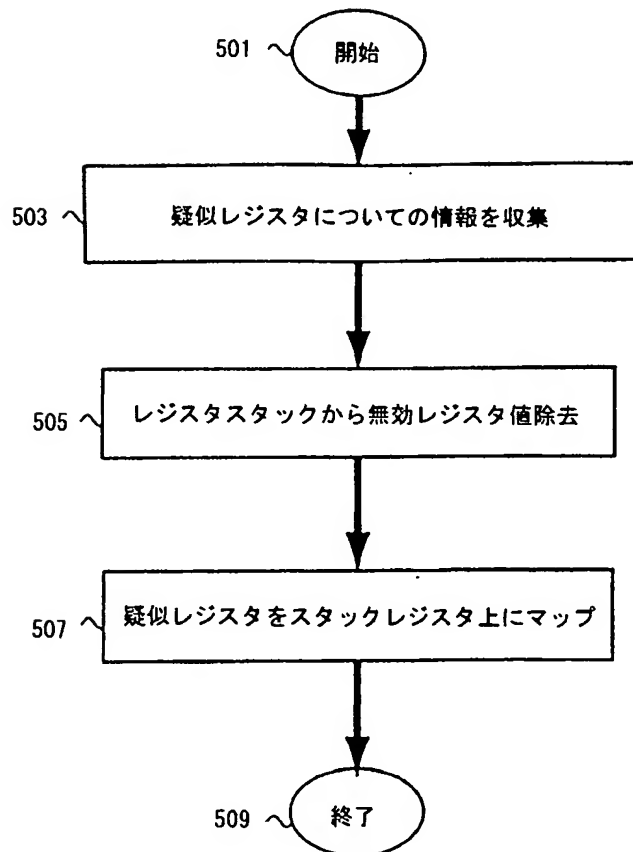


300 →

【図4】

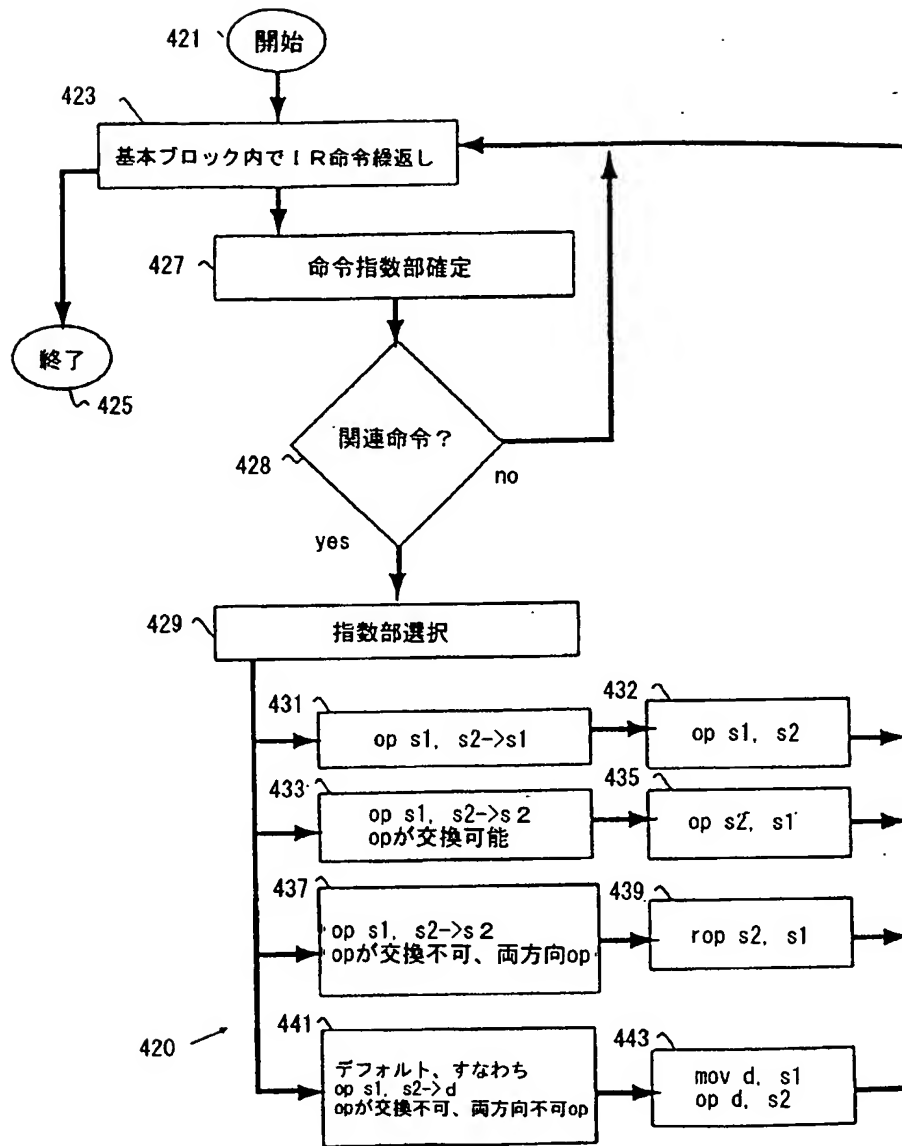


【図6】

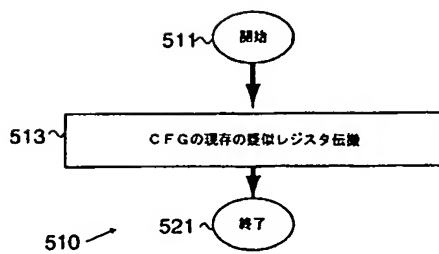


500 →

【図5】

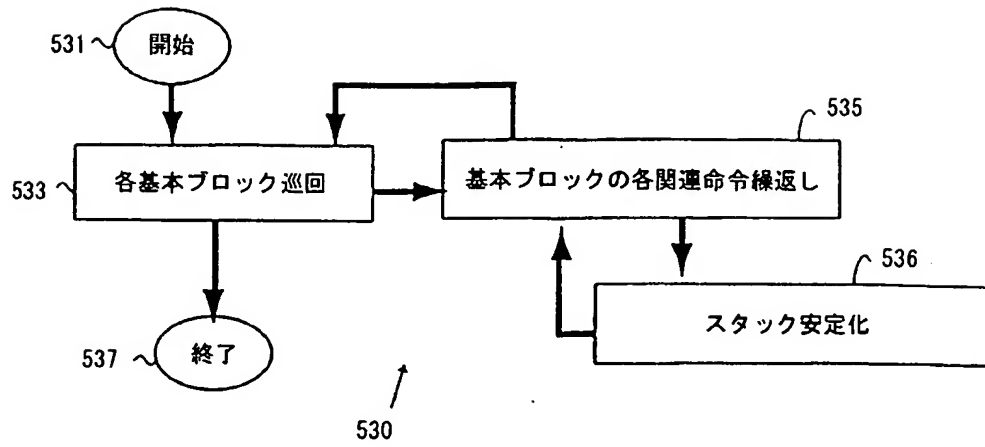


【図7】

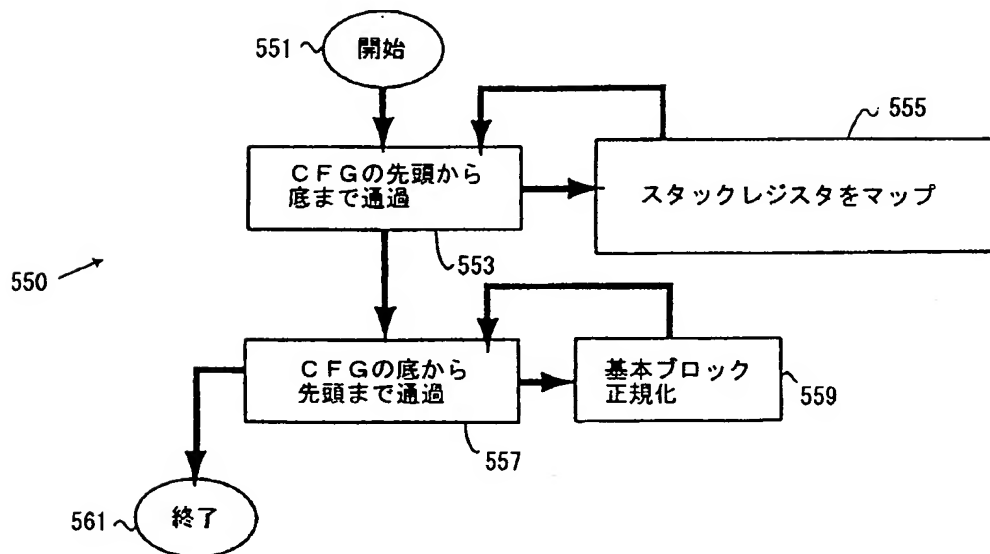




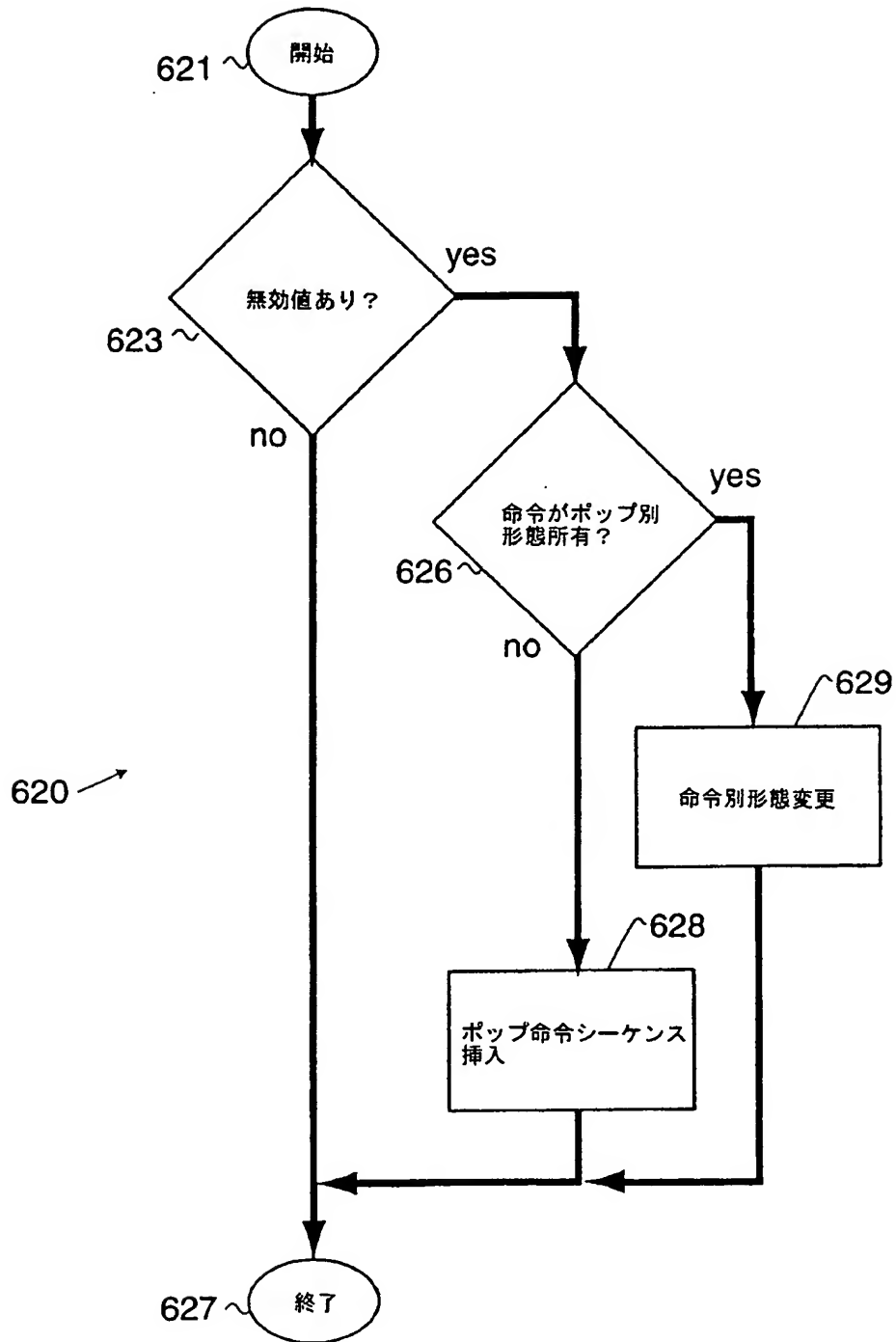
【図8】



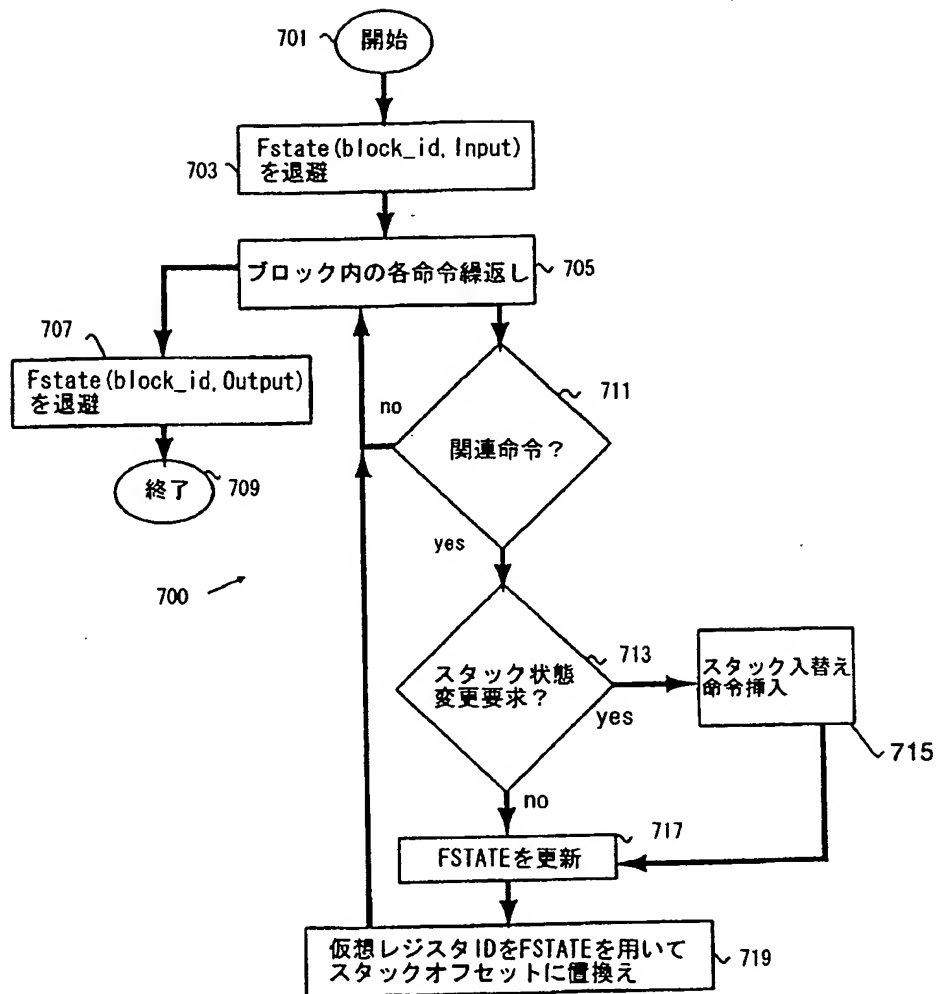
【図9】



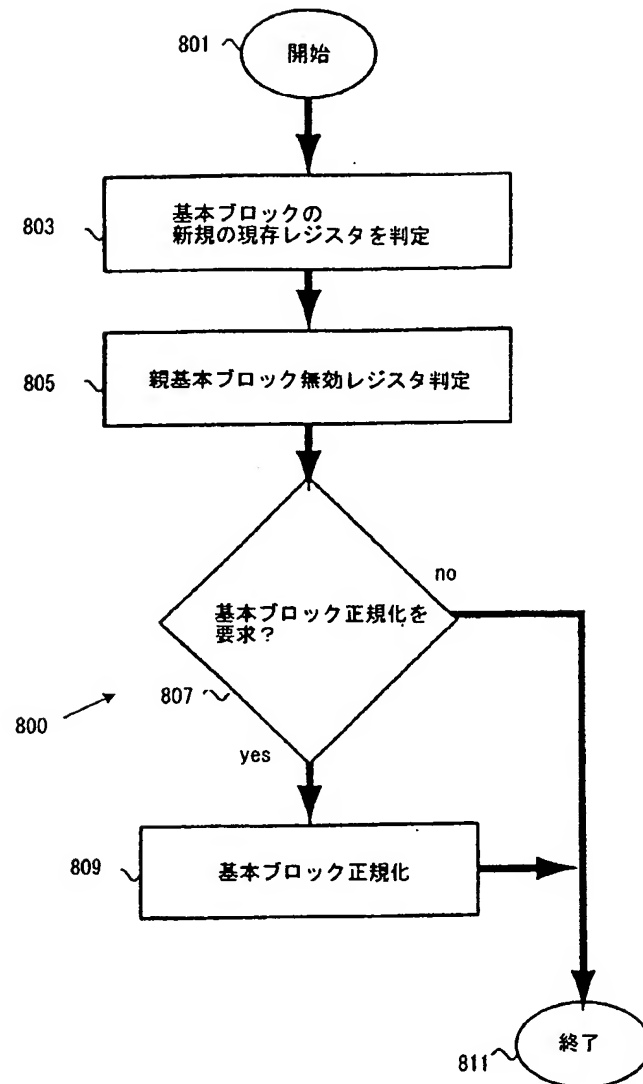
【図10】



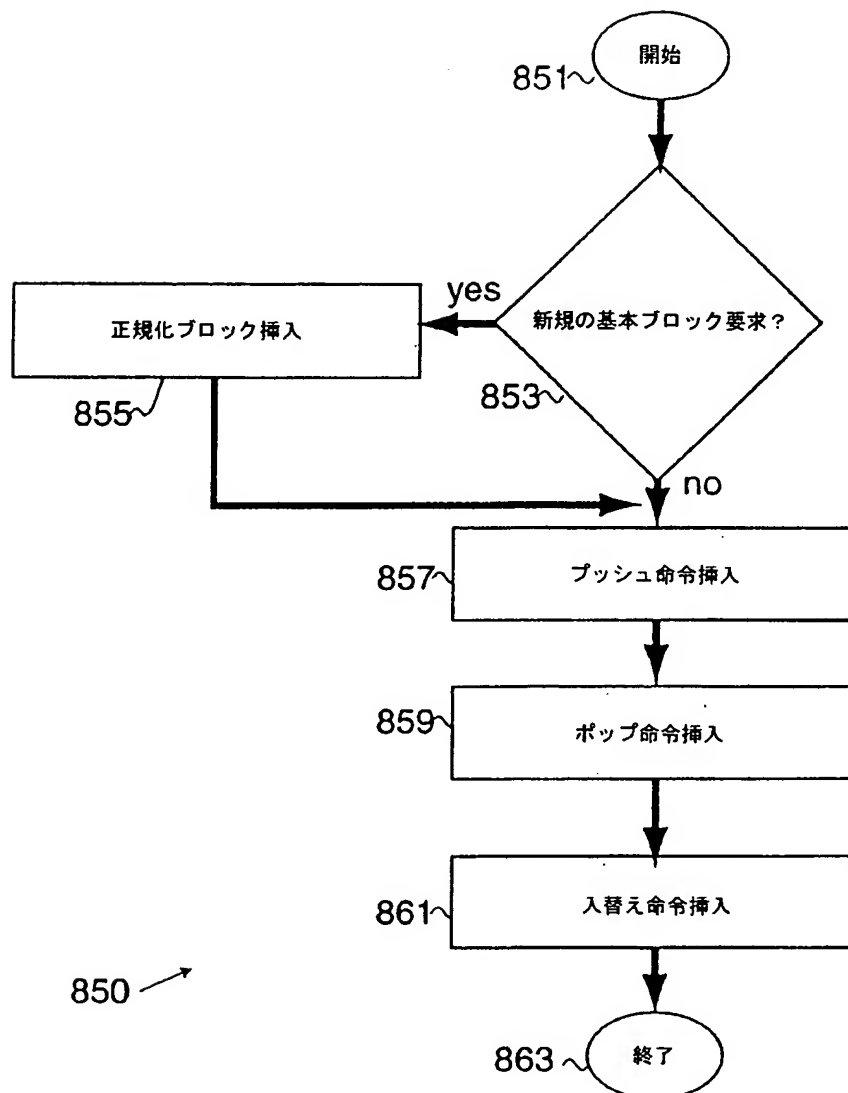
【図11】



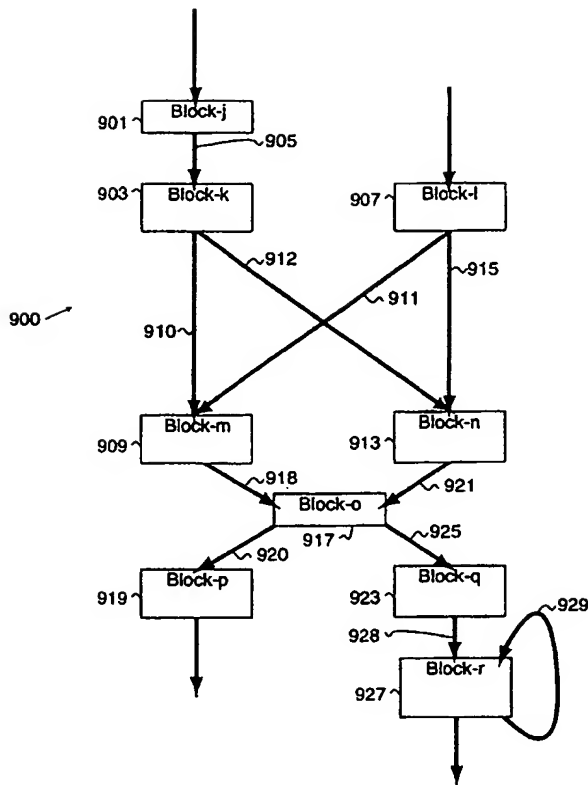
【図12】



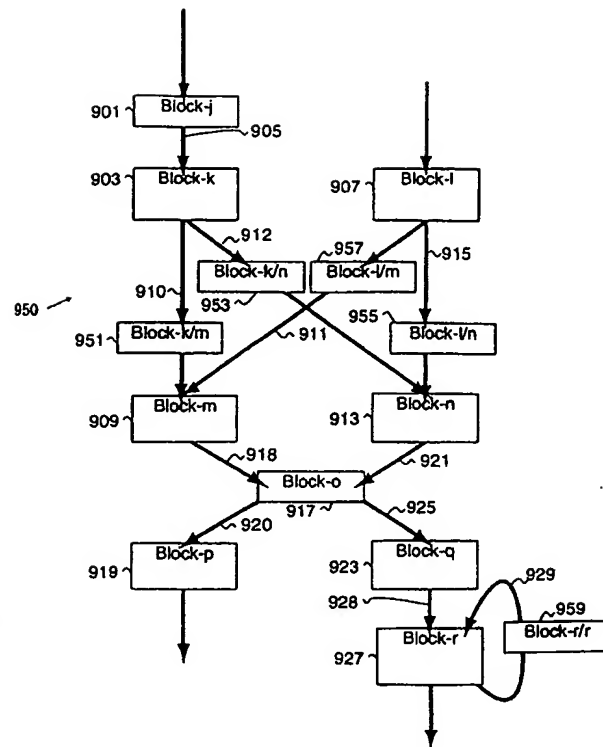
【図13】



【図14】



【図15】



フロントページの続き

(72)発明者 セルゲイ・ブイ・モロゾフ  
ロシア630057ノボシビルスク・232・ペチ  
ャトニコフストリート 9

(72)発明者 デビッド・エイ・セバーガー  
アメリカ合衆国カリフォルニア州94550・  
リバーモア・シャルドネーウェイ 2271

(72)発明者 デビッド・アール・ウォレス  
アメリカ合衆国カリフォルニア州94133・  
サンフランシスコ・ジョーンズストリート  
1960

(72)発明者 セルゲイ・エル・ウェニツキー  
ロシア630057ノボシビルスク・57・ペチ  
ャトニコフストリート 9

**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

**BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

☐ BLACK BORDERS

☒ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES

☒ FADED TEXT OR DRAWING

☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING

☐ SKEWED/SLANTED IMAGES

☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS

☐ GRAY SCALE DOCUMENTS

☒ LINES OR MARKS ON ORIGINAL DOCUMENT

☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY

☐ OTHER: \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**